1.Kernel

Ядро операційної системи — це центральний компонент операційної системи, який безпосередньо взаємодіє з апаратним забезпеченням комп'ютера та надає основні послуги іншим частинам системи та програмам.

Ядро відіграє ключову роль в управлінні ресурсами комп'ютера, такими як процесор, пам'ять, пристрої введення/виведення тощо. Всі програми і користувачі взаємодіють з ядром через системні виклики (системні функції), через які передаються запити до апаратного забезпечення.

Основні функції ядра операційної системи:

1. **Управління процесами**:
   - Ядро контролює створення, виконання та завершення процесів.
   - Воно визначає, які процеси мають право використовувати процесор в кожен момент часу (через планування процесів).
2. **Управління пам'яттю**:
   - Ядро управляє оперативною пам'яттю (RAM) і відповідає за її розподіл між процесами.
   - Воно також відповідає за віртуальну пам'ять, яка дозволяє програмам працювати навіть з більшими об'ємами даних, ніж доступно в оперативній пам'яті.
3. **Управління пристроями вводу/виводу (I/O)**:
   - Ядро взаємодіє з периферійними пристроями (наприклад, клавіатурою, мишею, принтерами, жорсткими дисками) через драйвери пристроїв.
   - Ядро забезпечує абстракцію апаратних пристроїв, що дозволяє програмам використовувати їх без необхідності знати конкретні деталі роботи з апаратним забезпеченням.
4. **Управління файловою системою**:
   - Ядро відповідає за збереження даних на дисках, управління файлами та каталогами.
   - Воно надає системні виклики для роботи з файлами (відкриття, читання, запис, видалення).
5. **Безпека та доступ**:
   - Ядро забезпечує безпеку системи, здійснюючи контроль доступу до ресурсів.
   - Воно також управляє правами користувачів і процесів, що забезпечує ізоляцію і запобігає несанкціонованому доступу.

Типи ядер:

1. **Монолітне ядро (Monolithic Kernel)**:
   - Це ядро, яке містить усі необхідні компоненти операційної системи в єдиному великому блоці коду. Воно обробляє всі основні операції: управління пам'яттю, планування процесів, взаємодію з пристроями тощо.
   - Приклад: **Linux**, **Unix**.
2. **Мікроядро (Microkernel)**:
   - Це ядро, яке намагається мінімізувати кількість функцій, що виконуються на рівні ядра. Тільки найосновніші функції, такі як планування процесів, управління пам'яттю та міжпроцесна комунікація, залишаються в ядрі. Інші сервіси виконуються в окремих процесах користувача.
   - Приклад: **Minix**, **QNX**.
3. **Гібридне ядро (Hybrid Kernel)**:

- Це комбіноване рішення, яке поєднує риси як монолітного, так і мікроядра. Воно включає більше функцій у ядрі, ніж мікроядро, але все ж намагається зберігати деякі сервіси в простір користувача.
- Приклад: **Windows NT**, **macOS**.

Детальніше розберемо монолітне ядро оскільки на ньому базується Linux

Основні риси монолітного ядра:

1. **Всі функції в одному блоці коду**: У монолітному ядрі всі компоненти операційної системи, такі як управління пам'яттю, процесами, файловою системою, драйверами пристроїв, виконуються в одному просторі — в режимі ядра (kernel space). Це забезпечує високу швидкість обміну інформацією між цими компонентами, оскільки вони не потребують складної комунікації між собою (як у випадку мікроядра, де багато сервісів виконуються в просторі користувача).
2. **Продуктивність та швидкість**: Через те, що більшість компонентів ядра працюють в одному просторі (в режимі ядра), монолітне ядро зазвичай забезпечує високу продуктивність і швидкість операцій, оскільки немає необхідності в контекстних перемиканнях між простором ядра та простором користувача, що характерно для мікроядер.
3. **Тісна взаємодія між компонентами**: Всі частини ядра безпосередньо взаємодіють одна з одною без необхідності в посередниках. Наприклад, драйвери пристроїв можуть безпосередньо звертатися до управління пам'яттю, а планувальник процесів може оперативно взаємодіяти з файловою системою без великих накладних витрат.
4. **Надійність і стабільність**: Оскільки всі компоненти ядра є частинами одного блоку, у разі помилки в одному з компонентів, вся система може вийти з ладу. Тому стабільність таких систем може залежати від якості коду ядра.

Основні компоненти монолітного ядра:

- **Управління процесами**: ядро монолітної операційної системи займається створенням, управлінням та завершенням процесів. Всі процеси виконуються в контексті користувача або в режимі ядра.
- **Управління пам'яттю**: монолітне ядро керує як фізичною, так і віртуальною пам'яттю. Зазвичай ядро містить механізми для управління сторінками пам'яті (paging) та сегментами (segmentation), що дозволяє ефективно використовувати пам'ять на системах з багатьма програмами.
- **Управління вводу/виводу**: монолітне ядро безпосередньо керує взаємодією з апаратними пристроями через драйвери. Всі драйвери вбудовані безпосередньо в ядро, що дозволяє знижувати затримки в обробці вводу/виводу.
- **Файлова система**: монолітне ядро відповідає за доступ до файлів і даних на диску, включаючи організацію файлових систем, збереження даних на носіях та їхній захист.

Переваги монолітного ядра:

1. **Швидкість**: Завдяки тому, що всі компоненти працюють в єдиному просторі пам'яті (режим ядра), між ними відсутні витрати на контекстні перемикання. Це дає високу продуктивність системи.
2. **Простота в дизайні**: Монолітне ядро має єдину структуру, і для розробників це може бути простіше, ніж створювати мікроядро, де кожен компонент (наприклад, драйвери, файлові системи) повинні працювати як окремі процеси з власними обмеженнями на доступ до апаратного забезпечення.
3. **Менше накладних витрат**: Оскільки всі функції ядра реалізуються в одному просторі, обмін даними між ними не вимагає додаткових накладних витрат, що

властиво мікроядрам, де кожен компонент працює в окремому процесі та взаємодіє з іншими через повідомлення або виклики системних функцій.

Недоліки монолітного ядра:

1. **Складність у підтримці та розширенні**: Оскільки всі компоненти ядра зібрані в єдиному блоці, зміни або доповнення до однієї частини ядра можуть вплинути на інші компоненти. Це може ускладнити розробку, налагодження та оновлення ядра.
2. **Низька ізоляція компонентів**: У разі помилки або несправності в одному з компонентів ядра, це може призвести до збоїв у всій системі. Наприклад, якщо драйвер пристрою містить баг, це може викликати падіння всього ядра.
3. **Обмежена гнучкість**: У монолітному ядрі всі компоненти сильно взаємозалежні, що може обмежувати гнучкість системи, наприклад, у випадку оновлення або заміни однієї частини ядра без перезавантаження всього компонента.

2. Libraries

Бібліотеки операційних систем (ОС) — це зазвичай набори функцій, які надаються ОС для полегшення розробки програмного забезпечення. Вони дозволяють програмам взаємодіяти з ресурсами комп'ютера (наприклад, файловою системою, пам'яттю, процесами) без необхідності програмувати низькорівневі операції.

Ось як це виглядає на прикладі двох найбільш поширених операційних систем: **Linux** та **Windows**:

1. Linux:

У Linux існує величезна кількість бібліотек, які надаються через системи пакетного менеджменту, такі як `apt` або `yum`, і часто використовуються програмістами для взаємодії з ОС.

- **GNU C Library (glibc)**: це стандартна бібліотека C для Linux, яка надає базові функції для взаємодії з операційною системою, зокрема для управління пам'яттю, робота з файлами, мережами, потоками тощо.
- **libpthread**: бібліотека для роботи з багатозадачністю, що надає механізми для створення та управління потоками.
- **libc**: стандартна бібліотека для роботи з основними операціями, такими як ввід/вивід, управління пам'яттю.
- **libX11**: бібліотека для створення графічних інтерфейсів користувача у середовищі X Window System.
- **libcurl**: бібліотека для роботи з протоколами HTTP, FTP та іншими мережевими протоколами.
- **OpenSSL**: бібліотека для роботи з криптографією та безпекою.

У Linux програмісти часто використовують стандартні системні виклики через бібліотеки, аби взаємодіяти з ОС на низькому рівні.

2. Windows:

У Windows також існує низка бібліотек, які полегшують взаємодію з операційною системою.

- **Windows API (WinAPI)**: основна бібліотека для доступу до функцій ОС. Вона надає інтерфейси для роботи з графічним інтерфейсом користувача (GUI), файловою

системою, процесами, потоками та мережею. Основні компоненти WinAPI включають:
    - **Kernel32.dll** — для основних функцій ОС, таких як управління пам'яттю, процесами, потоками, файловою системою.
    - **User32.dll** — для роботи з графічним інтерфейсом користувача.
    - **Gdi32.dll** — для роботи з графікою (малювання, шрифти, виведення на екран).
    - **Advapi32.dll** — для роботи з реєстром Windows, безпекою, акаунтами користувачів.
    - **Ws2_32.dll** — для мережевих операцій через сокети (TCP/IP).
- **.NET Framework**: це набір бібліотек і класів, які дозволяють програмістам працювати з високорівневими операціями, такими як робота з базами даних, веб-сервісами, інтерфейсами користувача тощо. .NET включає в себе класичні бібліотеки, як-от `System.IO`, `System.Net`, `System.Threading` і так далі.
- **DirectX**: набір бібліотек для роботи з графікою, відео, а також для аудіо та введення в реальному часі. Використовується для розробки ігор і мультимедійних додатків.

3. Порівняння та загальні принципи:

Обидві операційні системи надають програмістам бібліотеки для взаємодії з апаратним забезпеченням через абстракцію, що дозволяє програмам не залежати від конкретних деталей апаратної реалізації. Основні принципи, що лежать в основі таких бібліотек:

- **Системні виклики**: Функції, які надаються ОС для виконання операцій, як от робота з пам'яттю, файлами, процесами.
- **Абстракція**: Бібліотеки дозволяють програмістам працювати з високорівневими концепціями (наприклад, створення потоків або мережевих з'єднань), замість того щоб працювати з низькорівневими деталями апаратного забезпечення.
- **Портативність**: Деякі бібліотеки, як-от `glibc` або стандартна бібліотека C, забезпечують портативність програм між різними платформами.

Якщо тобі цікаві конкретні аспекти або приклади використання бібліотек ОС у програмуванні, я можу пояснити детальніше для певних мов програмування або бібліотек.

3. System utilities

Системні утиліти (або **системні інструменти**) — це програми або набори програм, які виконують допоміжні або обслуговуючі функції в операційній системі. Вони забезпечують ефективну роботу системи та дозволяють користувачам і адміністраторам комп'ютерів виконувати важливі операції з управління ресурсами, моніторингу, налаштування або діагностики системи.

Основні типи системних утиліт

1. **Утиліти для управління файлами та директоріями**:
    - **Файлові менеджери** — програми для організації та маніпуляції файлами (копіювання, переміщення, видалення, зміна прав доступу тощо).
        - **Windows**: Провідник (Explorer)
        - **Linux**: `ls`, `cp`, `mv`, `rm`, `find`, `tar`, `rsync`
    - **Архіватори** — програми для стиснення або розпакування файлів.
        - **Windows**: WinRAR, 7-Zip
        - **Linux**: `zip`, `tar`, `gzip`
2. **Утиліти для моніторингу та діагностики системи**:
    - **Моніторинг ресурсів** — програми для відстеження використання процесора, пам'яті, дискового простору та мережевого трафіку.

        - **Windows**: Диспетчер завдань (Task Manager), Performance Monitor
        - **Linux**: `top`, `htop`, `vmstat`, `iotop`, `free`
    - **Перевірка та виправлення помилок файлової системи**:
        - **Windows**: `chkdsk`
        - **Linux**: `fsck`
    - **Перевірка здоров'я жорстких дисків**:
        - **Windows**: CrystalDiskInfo
        - **Linux**: `smartctl` (з утиліти smartmontools)
3. **Утиліти для управління процесами**:
    - **Завдання та процеси** — програми для перегляду та управління запущеними процесами.
        - **Windows**: Диспетчер завдань (Task Manager)
        - **Linux**: `ps`, `top`, `kill`, `nice`
    - **Управління чергами завдань**:
        - **Windows**: Планувальник завдань (Task Scheduler)
        - **Linux**: `cron`, `at`
4. **Мережеві утиліти**:
    - **Діагностика мережі** — програми для тестування з'єднань, пінгування та трасування маршрутів.
        - **Windows**: `ping`, `ipconfig`, `tracert`, `netstat`
        - **Linux**: `ping`, `ifconfig`, `netstat`, `traceroute`, `nslookup`
    - **Моніторинг мережі**:
        - **Windows**: Resource Monitor (Монітор ресурсів)
        - **Linux**: `iftop`, `nmap`
5. **Утиліти для роботи з користувачами та безпекою**:
    - **Управління користувачами та правами доступу**:
        - **Windows**: Панель керування користувачами, `net user`
        - **Linux**: `useradd`, `usermod`, `chmod`, `chown`
    - **Інструменти для захисту та шифрування**:
        - **Windows**: BitLocker
        - **Linux**: `gpg`, `openssl`
6. **Системи резервного копіювання та відновлення**:
    - **Windows**: Бекап через Панель управління або інструменти на кшталт `wbadmin`
    - **Linux**: `rsync`, `tar`, `dd`
7. **Утиліти для налаштування системи**:
    - **Конфігурація мережі**:
        - **Windows**: Панель керування, `netsh`
        - **Linux**: `ifconfig`, `ip`, `nmcli`
    - **Управління службами та демонами**:
        - **Windows**: Служби (Services.msc)
        - **Linux**: `systemctl`, `service`
8. **Утиліти для обслуговування реєстру** (для Windows):
    - **Редактор реєстру (regedit)**: інструмент для редагування налаштувань реєстру.
    - **Пошук та видалення шкідливих програм**: Windows Defender, Malwarebytes.

Деякі приклади популярних системних утиліт:

1. **Windows**:
    - **Disk Cleanup** — для очищення жорсткого диска від тимчасових файлів.
    - **Task Scheduler** — для автоматизації виконання завдань на певний час.
    - **System Information** — для перегляду загальної інформації про систему.
    - **Device Manager** — для управління апаратним забезпеченням.
2. **Linux**:

- **`htop`** — розширена версія утиліти `top`, яка показує інформацію про ресурси в більш зручному вигляді.
- **`dmesg`** — виведення повідомлень ядра та інформації про запуск системи.
- **`lsof`** — показує список відкритих файлів і процесів.
- **`df`** — для перевірки простору на диску.

У Linux системі утиліти зазвичай зберігаються в кількох стандартних місцях на файловій системі, залежно від їх призначення, типу та контексту використання. Основні директорії для зберігання утиліт:

1. /bin — основні системні утиліти для користувачів

- **Значення**: Директорія `/bin` містить основні утиліти, необхідні для запуску системи та виконання базових операцій. Ці утиліти доступні для всіх користувачів і мають бути доступними навіть при завантаженні системи в "режимі тільки для читання".
- **Приклад утиліт**: `ls`, `cp`, `mv`, `cat`, `rm`, `echo`
- **Призначення**: Тут зберігаються базові утиліти, необхідні для роботи системи, а також для роботи в однокористувацькому режимі. Це основні команди, які використовуються для роботи з файлами, процесами, введенням/виведенням.

2. /sbin — системні утиліти для адміністраторів

- **Значення**: Директорія `/sbin` містить утиліти, які зазвичай використовуються системними адміністраторами (root) для управління системою. Вони часто використовуються для налаштування апаратного забезпечення, налаштування мережі, управління процесами, моніторингу системи тощо.
- **Приклад утиліт**: `ifconfig`, `shutdown`, `reboot`, `fsck`, `mount`, `iptables`
- **Призначення**: Утиліти в `/sbin` потрібні для адміністративних завдань, таких як налаштування мережі, управління файловими системами, перезавантаження або вимикання системи.

3. /usr/bin — загальні утиліти для користувачів

- **Значення**: Утиліти в `/usr/bin` використовуються для повсякденної роботи користувачів. Це найбільша кількість програм в системі, зокрема графічні додатки, текстові редактори, інструменти для програмування тощо.
- **Приклад утиліт**: `vim`, `nano`, `gcc`, `python`, `curl`, `git`, `wget`
- **Призначення**: Вся велика кількість утиліт та додатків для користувачів зберігається в `/usr/bin`. Це місце для більшості програм, які встановлюються як частина стандартної системи або сторонніми пакетами.

4. /usr/sbin — системні утиліти для адміністраторів (більш специфічні)

- **Значення**: Директорія `/usr/sbin` містить утиліти, які часто використовуються адміністраторами для специфічних завдань, які не є критичними для базового запуску системи, але потрібні для налаштування та обслуговування.
- **Приклад утиліт**: `apache2`, `sshd`, `networkd`, `useradd`
- **Призначення**: Утиліти для управління службами, мережами, користувачами, а також для налаштування більш специфічних компонентів операційної системи.

5. /bin та /sbin vs /usr/bin та /usr/sbin:

- У Linux зазвичай існує поділ між **/bin** та **/sbin** (утиліти для базового запуску системи) і **/usr/bin** та **/usr/sbin** (утиліти для загальних та адміністративних завдань). Однак у сучасних дистрибутивах часто здійснюється об’єднання цих директорій для зручності.
- Зазвичай **/usr/bin** і **/usr/sbin** містять утиліти, необхідні для повсякденної роботи системи, тоді як **/bin** і **/sbin** містять утиліти, необхідні для мінімальної функціональності, особливо в аварійних ситуаціях, коли система знаходиться в обмеженому режимі.

6. /lib та /libexec — бібліотеки та утиліти для виконання

- **/lib** містить основні бібліотеки, необхідні для роботи утиліт з директорій `/bin` і `/sbin`. Це бібліотеки, що підтримують функціонування основних програм та служб.
    - Приклад: `libc.so` (стандартна бібліотека C), бібліотеки для роботи з мережами, пристроями введення/виведення.
- **/libexec** містить утиліти, які використовуються іншими програмами (зазвичай не безпосередньо користувачем).
    - Приклад: утиліти для виконання внутрішніх операцій службами або програмами.

7. /opt — сторонні програми

- **Значення**: Директорія `/opt` використовується для зберігання сторонніх програм та пакетів, які не є частиною стандартного дистрибутива або основного репозиторію.
- **Приклад**: Програмне забезпечення, яке не входить в стандартний набір дистрибутива Linux, наприклад, програмне забезпечення для специфічних завдань, комерційні програми або сторонні пакети.

8. /usr/local/bin — локальні програми для користувачів

- **Значення**: Ця директорія містить утиліти, які було встановлено локально (зазвичай через джерела, а не через пакетний менеджер). Вона використовується для програм, які не є частиною стандартного набору системи, але мають бути доступні всім користувачам.
- **Приклад утиліт**: локально скомпільовані програми або пакети, наприклад, програми, скомпільовані з вихідного коду.

9. /usr/local/sbin — локальні системні утиліти

- **Значення**: Директорія для локальних утиліт адміністратора, які не є частиною стандартної системи, але потрібні для адміністрування і обслуговування.
- **Приклад**: Локальні версії утиліт для налаштування та адміністрування, скомпільовані або встановлені вручну.

Підсумок

Системні утиліти — це важливі інструменти для ефективного управління операційною системою, діагностики її стану та виконання необхідних операцій. Вони дозволяють як користувачам, так і адміністраторам зберігати стабільність, безпеку та оптимальну роботу комп'ютерів або серверів.