1. Поясніть етапи завантаження системи Linux.

1. Завантаження BIOS/UEFI

Після того, як комп’ютер увімкнено, на першому етапі запускається BIOS (Basic Input/Output System) або UEFI (Unified Extensible Firmware Interface) — в залежності від конфігурації системи. Це програмне забезпечення, вбудоване в материнську плату, яке:

- Ініціалізує апаратне забезпечення.
- Перевіряє підключені пристрої (оперативну пам'ять, диски, клавіатуру тощо).
- Завантажує завантажувач (bootloader) з пристрою, вказаного в налаштуваннях BIOS/UEFI (наприклад, жорсткого диска, SSD або USB).

2. Завантаження завантажувача (bootloader)

Завантажувач (наприклад, GRUB — Grand Unified Bootloader) відповідає за вибір операційної системи, якщо в системі встановлено кілька ОС, і за ініціалізацію ядра Linux. Етапи роботи завантажувача:

- Завантаження конфігурації, де вказано, яке ядро та параметри передати на наступному етапі.
- Після вибору ядра завантажувач передає керування системі.

3. Завантаження ядра Linux

Ядро Linux — це центральна частина операційної системи, яка відповідає за керування апаратними ресурсами (процесор, пам'ять, пристрої введення/виведення тощо). Завантаження ядра включає кілька важливих кроків:

- Завантаження ядра з диска в оперативну пам'ять.
- Ініціалізація основних драйверів для апаратного забезпечення, таких як контролери дисків, мережеві адаптери та інші пристрої.
- Виконання початкових налаштувань і підготовка середовища для запуску процесів користувача.
- Завантаження initial RAM disk (initrd або initramfs) — це тимчасова файлові система, що допомагає з початковою настройкою системи (зазвичай містить драйвери для завантаження, інструменти для роботи з дисками та інші необхідні утиліти).

4. Запуск процесу ініціалізації (init)

Після завантаження ядра, система переходить до запуску процесу ініціалізації. Цей процес запускається як перший процес у системі з PID 1 і відповідає за:

- Ініціалізацію системних служб (наприклад, монтування файлових систем, налаштування мережі).
- Запуск інших процесів, які необхідні для роботи операційної системи.

У Linux традиційно цей процес був `init`, але сучасні дистрибутиви часто використовують більш складні системи ініціалізації, такі як `systemd`.

5. Ініціалізація системи та запуск служб

На цьому етапі відбувається запуск основних служб і демони, необхідні для роботи системи:

- Завантаження драйверів і модулів ядра для роботи з різними пристроями.
- Монтування файлових систем, в тому числі root (кореневої) файлової системи.
- Налаштування мережі, запуск інтерфейсів.
- Запуск системних служб (наприклад, логування, обробка запитів на друк, підтримка зовнішніх пристроїв тощо).

6. Запуск користувацького середовища

Після того, як основні системні служби та демони запущені, система готова до використання:

- Якщо система працює в графічному режимі, запускається дисплейний менеджер (наприклад, `gdm`, `sddm`, `lightdm`), який відповідає за виведення екрану входу і аутентифікацію користувача.
- Якщо система запускається в текстовому режимі, користувач отримає командний рядок (термінал) для виконання команд.

7. Аутентифікація та вхід користувача

Після того, як система ініціалізована, користувач може увійти в систему. У графічному інтерфейсі це зазвичай відбувається через екран входу (login manager), в текстовому — через команду `login`.

8. Запуск користувацьких процесів

Після входу користувача в систему запускатимуться інші процеси, які зазвичай включають:

- Запуск оболонки (наприклад, `bash` або `zsh`).
- Завантаження конфігураційних файлів користувача.
- Запуск інструментів, необхідних для користувача, таких як графічні програми, мережеві утиліти і т. д.

2. Як переглянути системні журнали?

1.Перший спосіб:

1. cd var/log/

2. ls –a

3.Вибрати відповідний журнал який бажаєте переглянути (syslog, kern.log і тд.) за допомогою команди cat або less.

2.Другий спосіб:

1. journalctl та dmesg

3. -rw------- : Опишіть цей дозвіл. Як додати прапорець виконуваного файлу?

А) Дозвіл `-rw-------` вказує на права доступу до файлу або каталогу в системі, що використовує файлову систему, подібну до Unix/Linux.

Розбір цього дозволу:

- **Перший символ (`-`)**: Це тип файлу. У даному випадку це звичайний файл (не директорія чи спеціальний файл).
- **Три наступні символи (`rw-`)**: Права доступу для власника файлу (перші три символи після типу файлу).
    - `r` (read) — власник має право читати файл.
    - `w` (write) — власник має право записувати в файл (змінювати його).
    - `-` (без виконуваного доступу) — власник не має права виконувати файл як програму.
- **Три наступні символи (`---`)**: Права доступу для групи, до якої належить файл. У цьому випадку група не має жодних прав (не може ні читати, ні писати, ні виконувати файл).
- **Три останні символи (`---`)**: Права доступу для всіх інших користувачів (окрім власника та групи). У цьому випадку інші користувачі також не мають жодних прав на файл.

Таким чином, `-rw-------` означає, що:

- Власник може читати та змінювати файл, але не може його виконувати.
- Група та інші користувачі не мають жодного доступу до файлу.

Б)

1. За допомогою chmod u+x <назва_файлу> (a,g,o відповідно того кому надати доступ)

2. За допомогою chmod та чисел.

Ось як працюють ці числа:

- **Читання (r)** = 4
- **Запис (w)** = 2
- **Виконання (x)** = 1

Ці числа сумуються для кожної категорії користувачів (власник, група, інші). Наприклад:

- **7** = читання (4) + запис (2) + виконання (1) (тобто всі права).
- **6** = читання (4) + запис (2) (без виконання).
- **5** = читання (4) + виконання (1) (без запису).
- **4** = тільки читання.
- **3** = запис (2) + виконання (1) (без читання).
- **2** = тільки запис.
- **1** = тільки виконання.
- **0** = немає прав.

`apt` і `dpkg` — це два інструменти для управління пакетами в системах на базі Debian (включаючи Ubuntu), але вони працюють на різних рівнях і мають різні функціональні можливості.

4. Яка різниця між між apt і dpkg:

1. Рівень абстракції:

- **`dpkg`** — це низькорівневий інструмент для роботи з пакетами на рівні файлів. Він працює безпосередньо з `.deb` файлами, встановлюючи їх, видаляючи, або запитуючи інформацію про пакет.
- **`apt`** — це вищий рівень абстракції, який надає більш зручний інтерфейс для управління пакетами, включаючи автоматичну обробку залежностей, оновлення пакетів з репозиторіїв тощо.

2. Управління залежностями:

- **`dpkg`** не обробляє залежності між пакетами. Якщо ви намагаєтесь встановити пакет, який має невстановлені залежності, `dpkg` не зможе їх автоматично знайти або встановити.
    - Приклад: якщо ви намагаєтесь встановити пакет, але він має залежність від іншого пакету, вам доведеться вручну встановити всі необхідні залежності.
- **`apt`** автоматично керує залежностями. Якщо ви використовуєте `apt` для встановлення пакета, він завжди перевірить і автоматично встановить всі необхідні залежності.
    - Приклад: `apt install <назва_пакету>` автоматично завантажить і встановить всі залежності для цього пакету.

3. Джерела пакетів:

- **`dpkg`** працює лише з `.deb` файлами, які ви завантажуєте або отримуєте вручну.
    - Приклад: `dpkg -i <назва_пакету.deb>` — це команда для установки пакету з локального `.deb` файлу.
- **`apt`** працює з пакетами, що знаходяться в репозиторіях, зазначених у файлах конфігурації (наприклад, `/etc/apt/sources.list`). За допомогою `apt` можна встановлювати, оновлювати та видаляти пакети з віддалених репозиторіїв.
    - Приклад: `apt update` — оновлює список пакетів із репозиторіїв, а `apt upgrade` — оновлює всі встановлені пакети до останніх версій з репозиторіїв.

4. Інтерфейс:

- **`dpkg`** має простий інтерфейс командного рядка, який дає базові можливості для роботи з окремими пакетами (встановлення, видалення, перевірка статусу).
- **`apt`** надає більш зручний та комплексний інтерфейс для управління пакетами, включаючи можливості для оновлення системи, пошуку пакетів, вирішення проблем із залежностями тощо.