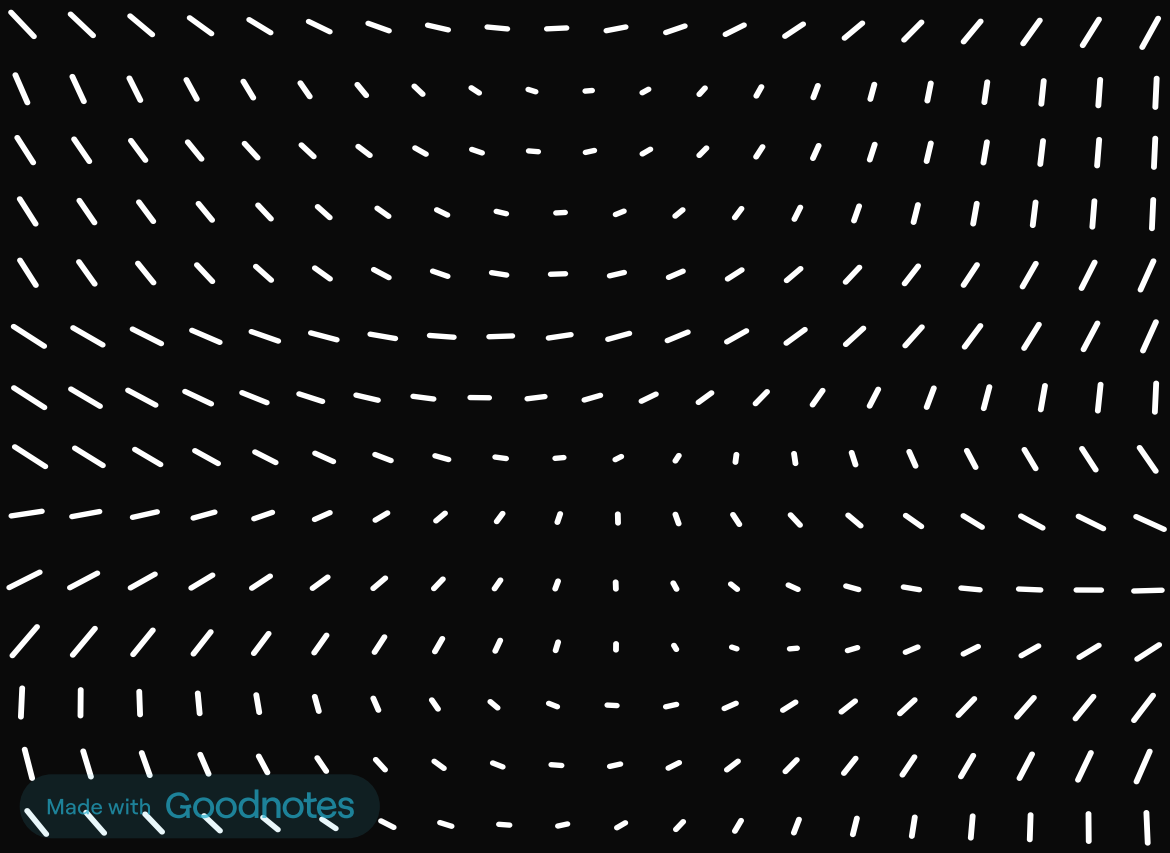
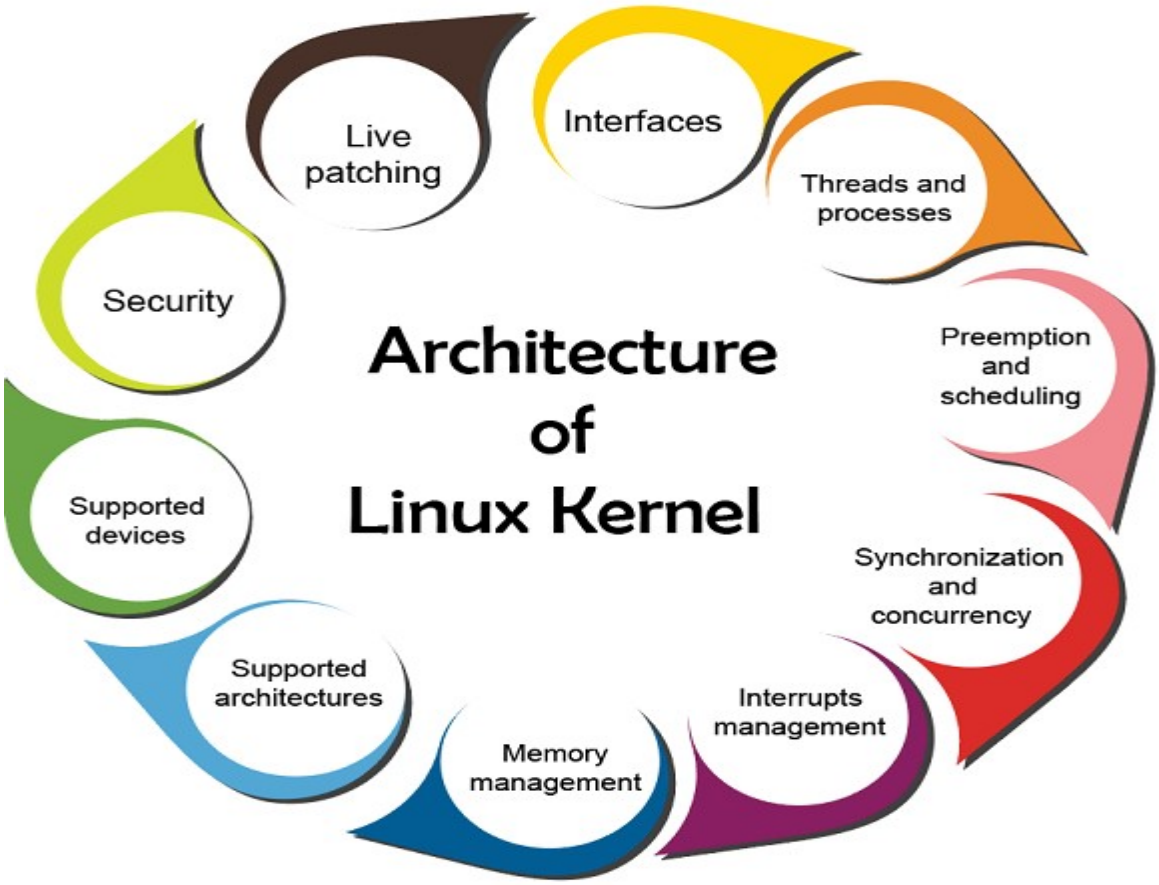
Linux Lecture 2



Linux Architecture: Kernel



1. Monolithic Kernel: (It is efficient but not modular)

- The Linux kernel is a monolithic kernel, which means that it contains most of the operating system's core functionalities within a single, unified structure.

- This design choice offers efficiency but reduces modularity.

2. Process Management:

- The kernel handles process creation, scheduling, and termination.

- It maintains a process table, which contains essential information about running processes.

3. Memory Management:

- The kernel manages physical and virtual memory, ensuring memory protection, paging, and swapping.

- The Memory Management Unit (MMU) plays a critical role in this process.

4. Device Drivers:

- Device drivers are integrated directly into the kernel or provided as loadable modules.

- These drivers enable communication between the hardware and software.

5. File System Management:

- The kernel provides access to the file system, which includes file I/O operations, file permissions, and maintaining file metadata (e.g., ownership, timestamps).

6. Networking:

- The kernel handles network protocols and supports various network devices and configurations.

- It is responsible for routing, packet handling, and network stack management.

7. Security:

- The kernel enforces security policies, such as user and group permissions, Access Control Lists (ACLs), and firewall rules.

- It ensures process isolation and data protection.

pipe → parent & child process comm
socket → comm b/w diff processes on a network
msg queues → comm by sending & receiving msg pkts in a queue

8. Inter-Process Communication (IPC):

- The kernel provides mechanisms for processes to communicate, including pipes, sockets, and message queues.

- These IPC methods facilitate collaboration between applications.

Architectural Differences between Windows and Linux :

1. Kernel Type:

- Windows uses a hybrid kernel, which combines features of both monolithic and microkernel architectures, resulting in complex interactions between components.

- Linux employs a pure monolithic kernel, offering more straightforward communication between kernel modules.

2. Licensing Model:

- Windows is a proprietary operating system, requiring users to purchase licenses for most versions.

- Linux is open-source, making it freely available, with various distributions to choose from.

3. GUI and Desktop Environment:

- Windows includes a tightly integrated graphical user interface (GUI), and the GUI is a fundamental part of the OS.

- Linux separates the GUI from the core OS, allowing users to choose from various desktop environments (e.g., GNOME, KDE, Xfce).

4. Software Installation and Package Management:

- Windows primarily relies on executable installers (e.g., MSI, EXE) for software installation.

- Linux uses package managers (e.g., APT, Yum, DNF) to install and manage software, simplifying updates and dependencies.

5. File System:

- Windows predominantly uses the NTFS file system, which is optimized for its platform.

- Linux supports multiple file systems (e.g., ext4, Btrfs, XFS), allowing users to choose the one that best suits their needs.

6. User Permissions and Security Model:

- Windows uses access control lists (ACLs) and a user-based permission model, which can sometimes be complex.

- Linux employs a more straightforward user and group permission model, making it easier to manage access and control.

7. Command Line Interface (CLI):

- Windows offers a command-line interface through PowerShell and Command Prompt, which are not as extensive as the Linux Terminal.

- Linux provides a robust command-line environment with a vast array of tools and scripting capabilities.

Configuration & Customizations of Linux :

1. Package Management:

- Use package managers like `apt`, `yum`, or `pacman` to install, update, and remove software.

- Customize software sources and repositories to control where packages are fetched from.

2. Shell Customization:

- Modify shell profiles (e.g., `.bashrc`, `.zshrc`) to tailor the command-line environment.

- Personalize the shell prompt, set environment variables, and create aliases for frequently used commands.

3. Kernel Configuration:

- Adjust kernel parameters using configuration files in `/etc/sysctl.conf` or by directly editing the `/proc` filesystem.

- Recompile the kernel or load/unload kernel modules to optimize performance or enable specific features.

4. User and Group Management:

- Create, modify, and manage user accounts and groups using commands like `useradd`, `usermod`, and `groupadd`.

- Set user permissions, create home directories, and manage group memberships.

5. Network Configuration:

- Configure network settings through files in `/etc/network/` or `/etc/sysconfig/network-scripts/` depending on the distribution.

- Adjust network interfaces, set IP addresses, and create custom routing rules.

6. File System Customization:

- Mount and manage file systems, specifying mount options and file system types in `/etc/fstab`.

- Create custom partitions, format disks, and configure quotas.

7. Security Hardening:

- Enhance system security by configuring firewalls (e.g., `iptables` or `ufw`) and intrusion detection systems (e.g., `Fail2ban`).

- Regularly update the system using the package manager to patch security vulnerabilities.

Linux Structure and Installation :

1. Filesystem Hierarchy:

- Linux follows a well-defined hierarchy with key directories, including `/bin`, `/etc`, `/home`, and `/var`.

- `/bin` contains essential binaries, `/etc` holds configuration files, `/home` hosts user home directories, and `/var` stores variable data.

UEFI – Unified Extensible Firmware Interface

2. Boot Process:

- The boot process begins with BIOS/UEFI firmware, which loads a bootloader (e.g., GRUB).

- The bootloader loads the Linux kernel, which initializes the system and launches essential services and daemons.

BIOS – Basic Input/Output Stream

3. Installation: GRUB - Grand Unified Bootloader

- The Linux installation process can vary between distributions but usually involves booting from installation media (USB, CD/DVD).

- During installation, users select partitions, specify file systems, choose software packages, and create user accounts as needed.

4. Package Management:

- After installation, package managers such as `apt`, `yum`, or `pacman` are used to update, install, or remove software packages.

5. Configuration:

- Post-installation configuration includes setting up user accounts, network settings, and other system-specific configurations.

6. Updates:

- Regularly update the system using the package manager to ensure the latest security patches and software updates are applied.

Reference

https://www.geeksforgeeks.org/introduction-to-linux-operating-system/

https://www.geeksforgeeks.org/the-linux-kernel/