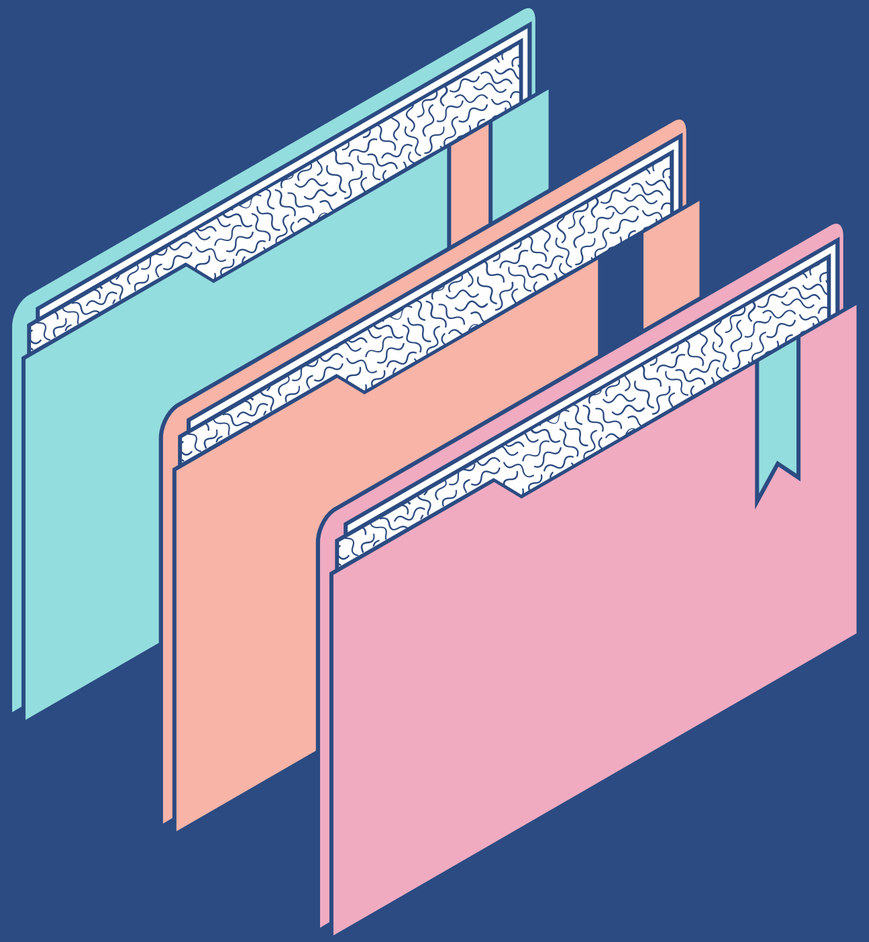
Unveiling the Power of Debian and Ubuntu: A Deep Dive into Open-Source Operating Systems



Agenda

KEY TOPICS DISCUSSED IN
THIS PRESENTATION

- Brief Explanation of OS
- Architecture of OS
- Process Control Mechanism
- Scheduling Algorithms
- Salient Features

Introduction

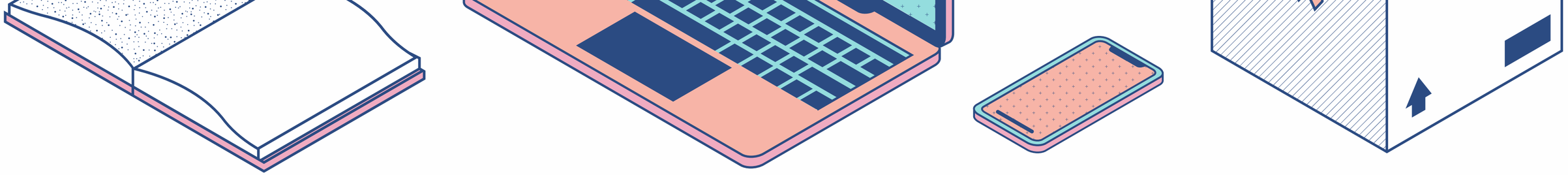
WHAT IS LINUX?

Linux is a free and open-source operating system kernel that serves as the foundation for a variety of Unix-like operating systems.

- The code used to create Linux is free and available to the public to view, edit, and for users with the appropriate skills—to contribute to

Linux has many distributions or “Distros”. Some of them are –

- Debian
- Ubuntu
- Fedora
- RedHat
- Cent OS
- openSUSE
- Arch LINUX
- Gentoo, and hundreds more



Brief Explanation of OS

DEBIAN

- Debian is a free and open-source Unix-like operating system and Linux distribution.
- It is known for its stability, reliability, and commitment to free software principles.
- It is developed by a global community of volunteers.

UBUNTU

- Ubuntu is a popular Linux distribution based on Debian.
- It is known for its ease of use, extensive software library, regular release cycle, and strong community support.
- Ubuntu is widely used for desktop, server, cloud, and IoT (Internet of Things) deployments.

ABOUT DEBIAN

- **Free & Open-Source:** Debian is a free operating system, just like Windows or macOS, but built with open-source software. This means you can use it for free and even modify its code if you choose to.
- **Linux Distribution:** Debian is a specific version of Linux, a powerful and stable operating system kernel.
- **Old & Reliable:** Launched in 1993, Debian is one of the oldest and most respected Linux distributions known for its stability and focus on security.
- **Huge Software Selection:** Debian offers a massive repository of over 59,000 software packages, giving you a vast selection of programs to choose from.
- **Community-Driven:** Debian is developed and maintained by a large, collaborative community of volunteers.Debian comes in various versions, providing multiple editions.

UI:


DEBIAN DESKTOP
All Applications
Accessories
Education
Games
Graphics
Internet
Office
Other
Sound & Video
Administration
Preferences
Places
Accessibility
Account details
Applets
Archive Manager
Backgrounds
Bluetooth Manager
Brasero
Calculator
Cheese
Chess
Color
Date & Time
Menu
10:17

ABOUT UBUNTU

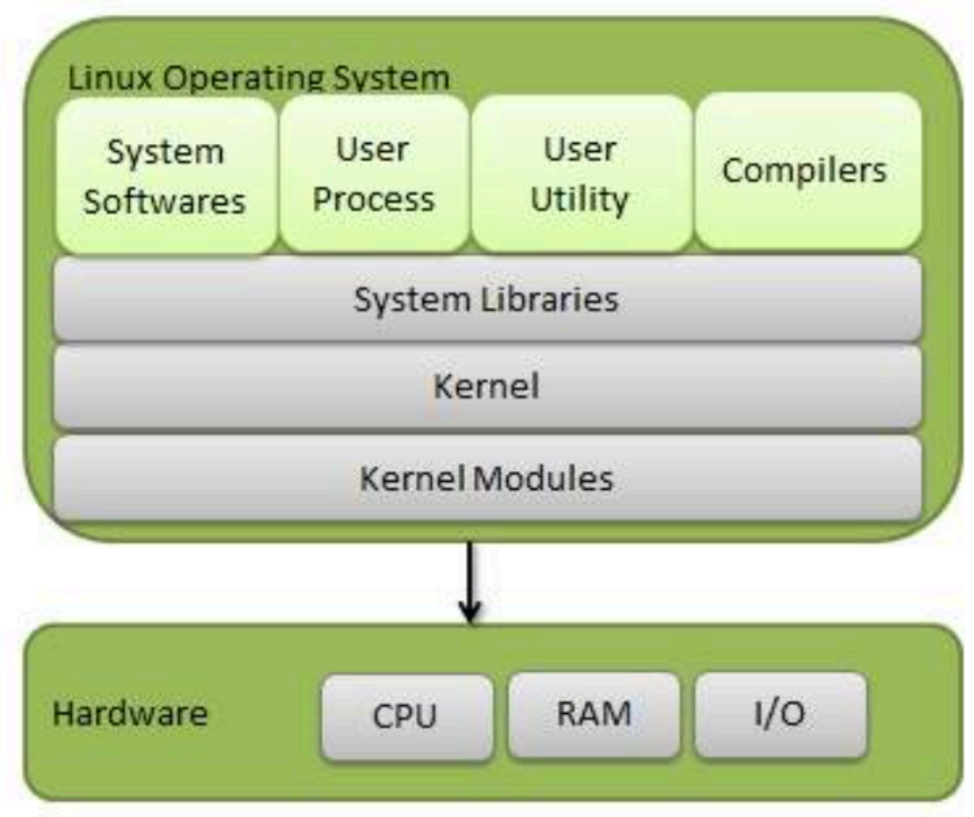
- **Free & Open-Source OS:** Think of it as a free operating system like Windows or macOS, but built on the Linux foundation. You can freely use and even modify Ubuntu's code.
- **User-Friendly Linux:** While Linux is known for power users, Ubuntu offers a friendly interface for beginners, making Linux more approachable.
- **Secure and Stable:** Inheriting from Linux, Ubuntu boasts strong security features and a reputation for stability.
- **Multiple Editions:** It caters to various needs with dedicated versions for desktops, servers, and even internet-connected devices.

UI:


UBUNTU DESKTOP
Activities
Sun Jun 3, 17:04
Type to search...
Boxes
Calculator
Cheese
Configure Fla...
Files
FileZilla
Firefox Web B...
GNU Image M...
Google Chrome
Google Keep - ...
HandBrake
HydraPaper
Input Method
Kazam
Language Sup...
Launch Flames...
LibreOffice 6.0
LibreOffice 6....
LibreOffice 6....
LibreOffice 6....
LibreOffice 6....
LibreOffice 6....
LibreOffice 6....
mpv Media Pla...
Frequent
All

Architecture of OS

Ubuntu builds on the Debian architecture and infrastructure and collaborates widely with Debian developers, but there are important differences. Ubuntu has a distinctive user interface, a separate developer community.

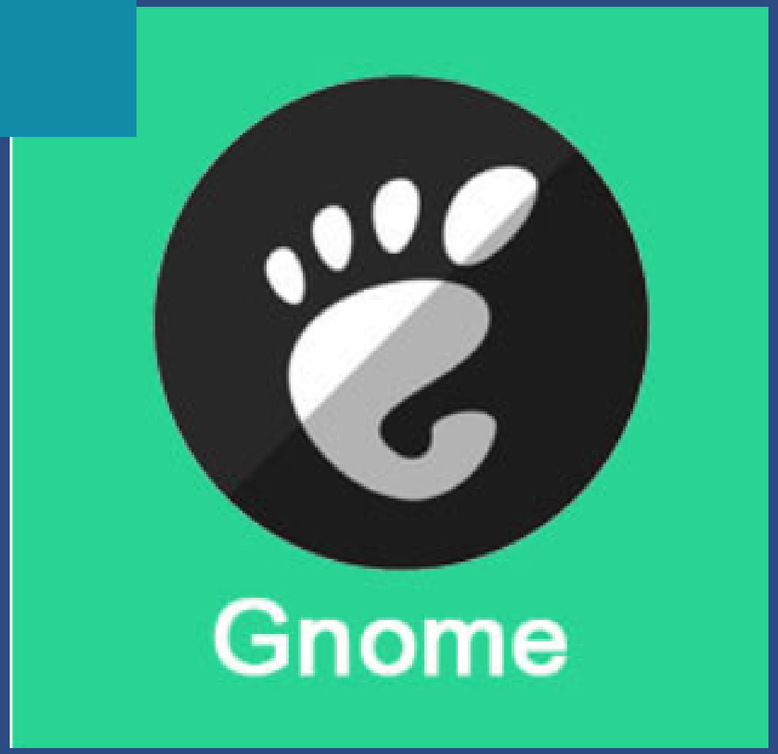


Debian Architecture:

- Debian is one of the oldest and most influential Linux distributions.
- It's known for its stability, robustness, and adherence to the principles of free and open-source software.
- Debian follows a modular architecture where components are designed to work together cohesively. The package management system, APT (Advanced Package Tool), is central to Debian's architecture.
- Debian uses the .deb package format for software installation and management.

Ubuntu Architecture:

- Ubuntu is based on Debian and shares many architectural principles with it. However, Ubuntu aims to provide a more user-friendly and polished experience out of the box.
- Like Debian, Ubuntu utilizes the APT package management system and .deb package format.
- Ubuntu has its own software repositories, which are based on Debian's but often include newer versions of software packages
- Ubuntu has its own desktop environment called Unity and now GNOME, though it supports various other desktop environments.



PROCESS CONTROL MECHANISMS

PROCESS SCHEDULING

INTER-PROCESS COMMUNICATION

ERROR HANDLING AND RECOVERY

CONCURRENCY CONTROL

CONTEXT SWITCHING

PRIORITY MANAGEMENT

PROCESS CONTROL BLOCK

1. **Process ID (PID): Unique identifier for the process.**
2. **Process State: Current state of the process (e.g., running, ready, blocked).**
3. **Program Counter (PC): Memory address of the next instruction to be executed.**
4. **CPU Registers: Values of CPU registers associated with the process.**
5. **Memory Management Information: Details about the process's memory usage.**
6. **File Descriptors: Tracks open files, sockets, and other I/O resources.**
7. **Process Priority: Scheduling priority of the process.**
8. **Parent Process ID (PPID): Identifier of the parent process.**

Process Control Block

Process creation...

In Ubuntu, as in any Unix-like operating system, processes can be created using several methods, with forking and spawning being the most common.

Forking

- Forking is a fundamental mechanism for process creation in Unix-like systems.
- The fork() system call is used, which creates a new process (child process) as a copy of the calling process (parent process).

Spawning

- Libraries like posix_spawn() provide an alternative to using fork() and exec() directly, offering a more convenient way to create processes with specific attributes.
- Spawning allows for more control over the process creation process, such as setting attributes like file descriptors, environment variables, and signal handlers.

Here's an example of how to CREATE a task in UBUNTU...

```c
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid;

    pid = fork(); // Create a new process

    if (pid < 0) {
        // Error handling
        fprintf(stderr, "Fork failed\n");
        return 1;
    } else if (pid == 0) {
        // Child process
        printf("Child process (PID: %d) created\n", getpid());
        // Child process logic here
    } else {
        // Parent process
        printf("Parent process (PID: %d) created a child (PID: %d)\n", getpid(), pid)
        // Parent process logic he
    }
```

Process termination..

Voluntary Termination:

Exit System Call: Processes can voluntarily terminate themselves by calling the exit() system call.

Involuntary Termination:

- Signal Termination: Processes can be involuntarily terminated by receiving signals. Signals are software interrupts sent to a process to notify it of an event or condition. For example:
- **SIGKILL:** Terminate the process immediately. This signal cannot be caught or ignored.
- **SIGTERM**: Terminate the process gracefully. The process can catch this signal and perform cleanup tasks before exiting.
- **SIGINT**: Terminate the process when the user presses **Ctrl+C in the terminal.**

Here's an example of how to TERMINATE a task in UBUNTU...

```c
void sigint_handler(int signum) {
    printf("Received SIGINT. Terminating...\n");
    exit(EXIT_SUCCESS);
}

int main() {
    // Register signal handler for SIGINT
    if (signal(SIGINT, sigint_handler) == SIG_ERR) {
        perror("signal");
        return EXIT_FAILURE;
    }

    printf("Running. Press Ctrl+C to terminate.\n");

    // Infinite loop to keep the process running
    while (1) {
        sleep(1);
    }

    return 0;
```

PROCESS SCHEDULING

The Scheduler:

- The Linux kernel, the core of Ubuntu, includes a process scheduler that makes these decisions.
- Scheduling is preemptive, meaning a higher priority process can interrupt a currently running process.

Scheduling Policies:

- Ubuntu uses the Completely Fair Scheduler (CFS) for most processes. CFS focuses on fairness by allocating CPU time slices to runnable processes in a round-robin fashion.
- Ubuntu supports real-time scheduling policies for processes requiring guaranteed responsiveness, like audio processing. These policies prioritize real-time tasks over normal processes.

PROCESS SCHEDULING

Scheduling Decisions:

- Process Priority: Higher priority processes generally get CPU time first. You can adjust process priority using the chrt command.
- Process Niceness: This allows you to voluntarily lower a process's priority, giving other processes more CPU time. You can adjust niceness using the nice command.
- Process State: Only runnable processes can be scheduled to run. Processes waiting for I/O or other resources are placed in wait queues.

CONCURRENCY CONTROL

CONCURRENCY CONTROL IN UBUNTU, AS IN ANY MODERN OPERATING SYSTEM, REFERS TO THE MECHANISMS AND TECHNIQUES USED TO MANAGE AND COORDINATE CONCURRENT ACCESS TO SHARED RESOURCES BY MULTIPLE PROCESSES OR THREADS.

Mutexes (Mutual Exclusion)

A mutex ensures only one thread can access a shared resource at a time. Other threads attempting to access the resource will be blocked until the mutex is released.

Semaphores

These act as generalized counters that control access to a limited number of resources. A semaphore can be used for scenarios where multiple threads need access to a fixed pool of resources, not just a single one. POSIX semaphore functions (sem_init(), sem_wait(), sem_post(), etc.).

Spinlocks

A busy-waiting mechanism where a thread trying to acquire a lock keeps trying (spinning) until it succeeds. Spinlocks can be faster than mutexes in specific situations but can consume CPU resources while waiting.

INTER-PROCESS COMMUNICATION

INTER-PROCESS COMMUNICATION (IPC) FACILITATES COMMUNICATION AND DATA EXCHANGE BETWEEN DIFFERENT PROCESSES RUNNING ON THE SYSTEM

Pipes

- Pipes provide a simple form of IPC for communication between two related processes, typically a parent process and its child.
- There are two types of pipes: unnamed pipes (created using the pipe() system call) and named pipes (also known as FIFOs, created using the mkfifo() system call).

Shared Memory

- This mechanism allows processes to share a contiguous memory segment, enabling direct data exchange. Processes can read and write to this shared memory region, providing faster communication compared to pipes. However, synchronization is crucial to avoid data corruption.

Message Queues

- Message queues allow processes to exchange messages in the form of structured data.
- In Ubuntu, message queues are implemented using POSIX message queues (mq_open(), mq_send(), mq_receive(), etc.).
- Message queues support both one-to-one and many-to-one communication patterns, where multiple processes can send messages to a single queue.

INTER-PROCESS COMMUNICATION

INTER-PROCESS COMMUNICATION (IPC) FACILITATES COMMUNICATION AND DATA EXCHANGE BETWEEN DIFFERENT PROCESSES RUNNING ON THE SYSTEM

Sockets

- Sockets are a more versatile IPC mechanism enabling communication across networks, not just on the same machine. Processes can establish connections using sockets and exchange data bidirectionally, making them suitable for client-server applications.

Signals

- Signals are software interrupts used to notify processes of events or asynchronous events.
- Ubuntu provides a set of predefined signals (e.g., SIGINT, SIGTERM, SIGKILL) that can be sent to processes using the kill() system call or raised by the kernel in response to specific events.

CONTEXT SWITCHING

Context switching in Ubuntu, as in any modern operating system, refers to the process of saving and restoring the state of a process or thread so that it can be temporarily paused and another process or thread can be executed.

Process and Thread States:

- Each process or thread in Ubuntu has its own execution context, which includes information such as CPU registers, program counter, stack pointer, and other relevant state.
- Processes and threads can be in various states, such as running, ready, blocked, or terminated, depending on their current execution status and resource requirements.

Triggering a Context Switch:

- Context switches are triggered by the operating system scheduler when it decides to switch execution from one process or thread to another.
- Context switches can occur for various reasons, including time slicing (preemptive multitasking), I/O operations, interrupts, or process/thread priorities.

CONTEXT SWITCHING

Context switching in Ubuntu, as in any modern operating system, refers to the process of saving and restoring the state of a process or thread so that it can be temporarily paused and another process or thread can be executed.

Saving the Current Context:

- Before switching to a new process or thread, the operating system saves the state of the currently executing process or thread.
- This involves storing the values of CPU registers, program counter, stack pointer, and other relevant state information into the process/thread control block (PCB or TCB).

Loading the New Context:

- Once the current context is saved, the operating system selects the next process or thread to execute based on its scheduling algorithm.
- The state of the selected process or thread, stored in its PCB/TCB, is then loaded back into the CPU registers, program counter, stack pointer, etc.

CONTEXT SWITCHING

Context switching in Ubuntu, as in any modern operating system, refers to the process of saving and restoring the state of a process or thread so that it can be temporarily paused and another process or thread can be executed.

Resuming Execution:

- With the new context loaded, execution resumes from the point where the previous process or thread was paused.
- The selected process or thread continues executing until it voluntarily yields the CPU (e.g., by blocking on I/O) or until the scheduler decides to switch to another process or thread.

ERROR HANDLING

Error handling and recovery in Ubuntu, like in any modern operating system, involves mechanisms and techniques to detect, report, and mitigate errors that occur during system operation

Error Reporting:

- Ubuntu provides mechanisms for applications and system components to report errors and exceptions. Common methods include logging errors to system logs (e.g., syslog or systemd journal), displaying error messages to users through graphical or command-line interfaces, and generating error reports for analysis

System Logs:

- Ubuntu maintains system logs that record information about system events, including errors, warnings, and other relevant messages.
- The syslog daemon (or systemd journal) collects and stores log messages from various system components, making it easier to diagnose and troubleshoot issues.

ERROR HANDLING

Error handling and recovery in Ubuntu, like in any modern operating system, involves mechanisms and techniques to detect, report, and mitigate errors that occur during system operation

Error Detection:

- Ubuntu continuously monitors system components and hardware devices for errors and anomalies. Various subsystems, drivers, and daemons are responsible for detecting errors and reporting them to the system log.
- Hardware monitoring tools, SMART diagnostics for storage devices, and system health monitoring utilities help detect hardware failures and performance issues.

Fault Tolerance:

- Ubuntu supports fault-tolerant configurations and techniques to minimize the impact of errors and failures on system operation.
- Redundant components, such as RAID arrays for storage, redundant power supplies, and clustering or load balancing for services, help ensure system resilience and availability in the event of failures.

PRIORITY MANAGEMENT

Priority management in Ubuntu involves assigning priorities to processes or threads to control their access to system resources

Process Priorities: Each process is assigned a "nice" value, determining its scheduling priority. Lower values represent higher priority.

Scheduler Priorities: The Completely Fair Scheduler (CFS) dynamically assigns priorities to processes based on their nice values and other factors, ensuring fair CPU allocation.

Scheduling Policies: Ubuntu supports time-sharing scheduling for interactive and batch processes, and real-time scheduling for time-sensitive tasks.

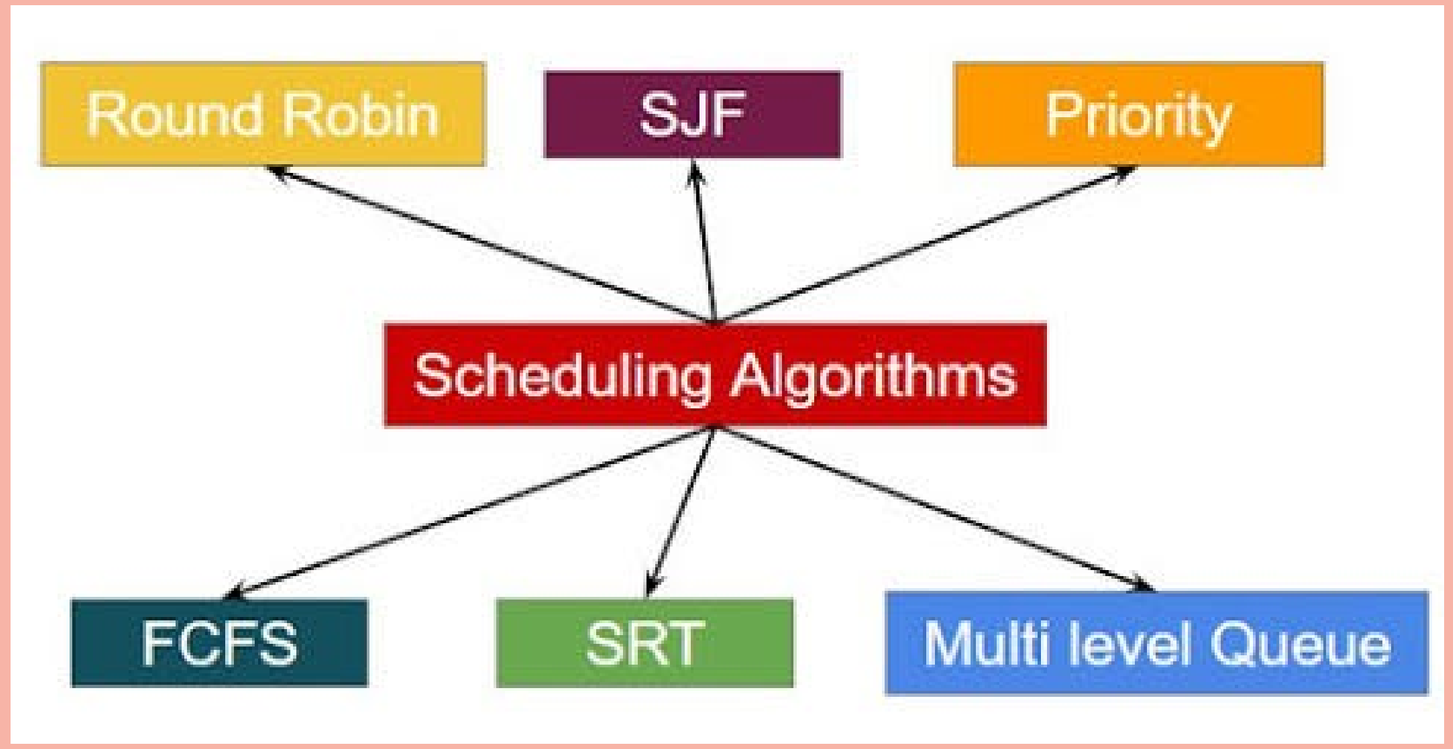
Priority Inversion: Ubuntu mitigates priority inversion through mechanisms like priority inheritance or donation to prevent high-priority processes from being blocked by low-priority ones.

Kernel Priorities: Kernel threads and interrupt service routines have priorities managed by the kernel scheduler to ensure efficient system operation.

Scheduling algorithms

"SELECTING A SCHEDULING ALGORITHM IS LIKE CRAFTING A RECIPE; IT'S ABOUT FINDING THE PERFECT BLEND OF INGREDIENTS TO ENSURE SMOOTH AND EFFICIENT OPERATION."

Both Ubuntu and Debian, being Linux-based operating systems, typically use the same or very similar scheduling algorithms. This is because both Ubuntu and Debian distributions use the Linux kernel as their core operating system. The Linux kernel provides various scheduling algorithms, and the default scheduler, such as the Completely Fair Scheduler (CFS), is commonly used across different Linux distributions, including Ubuntu and Debian.



Here's a list of commonly used scheduling algorithms in Debian/Ubuntu:

Completely Fair Scheduler (CFS):

- Default scheduler in the Linux kernel
- Dynamically assigns CPU time slices to processes based on their priority to achieve fairness.

Round Robin (RR):

- Each process is allocated a fixed time slice (quantum) to execute before being preempted.
- Helps in providing equitable CPU allocation, especially in time-sharing environments.

Earliest Deadline First (EDF):

- Real-time scheduling algorithm where tasks are scheduled based on their deadlines.
- Guarantees that the task with the earliest deadline will be executed first.

Shortest Job First (SJF):

- Selects the process with the shortest expected processing time for execution next.
- Typically used in non-preemptive scenarios and requires knowledge of process burst times.

Priority Scheduling:

- Each process is assigned a priority, and the scheduler selects the process with the highest priority for execution.
- Can be preemptive or non-preemptive.

Multi-level Queue Scheduling:

- Divides processes into multiple queues with different priority levels.
- Each queue may have its own scheduling algorithm.

Salient features

Ubuntu and Debian are two popular Linux distributions, each with its own set of features and characteristics. Here's a comparison of their salient features and how they contrast with other operating systems:

Ubuntu:

1. **User-Friendly:** Ubuntu focuses on ease of use, making it accessible to beginners. It comes with a graphical user interface (GUI) that resembles those of other mainstream operating systems, such as Windows and macOS.
2. **Regular Release Cycle:** Ubuntu follows a strict release schedule, with new versions coming out every six months. This ensures users have access to the latest features and improvements on a regular basis.
3. **Large Package Repository:** Ubuntu offers a vast repository of software packages, making it easy to find and install applications for various needs through its package management system, APT (Advanced Package Tool).
4. **Strong Community Support:** Ubuntu has a large and active community of users and developers who provide support, documentation, and assistance through forums, wikis, and other channels.
5. **Commercial Support:** Canonical, the company behind Ubuntu, offers commercial support options for businesses and organizations, including long-term support (LTS) releases with extended maintenance periods.

Debian:

1. **Stability:** Debian prioritizes stability and reliability over cutting-edge features. Its release cycle is more conservative compared to Ubuntu, with major releases occurring less frequently.
2. **Wide Hardware Support:** Debian supports a wide range of hardware architectures, making it suitable for various devices, including servers, desktops, and embedded systems.
3. **Community-Driven:** Debian is developed and maintained by a community of volunteers from around the world. It emphasizes principles such as open development, democratic decision-making, and transparency.
4. **Purely Free Software:** Debian is committed to promoting free and open-source software (FOSS). It adheres to strict guidelines regarding software licensing and only includes free software in its official repositories.
5. **Customizability:** Debian provides a high degree of flexibility and customizability, allowing users to configure their systems according to their specific requirements and preferences.

Comparison with other operating systems:

Windows

Compared to Windows, both Ubuntu and Debian are open-source and free to use. They offer more flexibility and customization options, but may have a steeper learning curve for users accustomed to the Windows environment.

macOS

While macOS is known for its user-friendly interface and seamless integration with Apple hardware, Ubuntu and Debian offer similar features with broader hardware support and more software customization options.

Other linux distributions

Ubuntu and Debian share many similarities with other Linux distributions, such as Fedora, CentOS, and openSUSE. However, each distribution has its own unique characteristics, package management systems, and target audiences.

Member contribution

TEAM 14

DEBIAN/UBUNTU

Aravinth	Vicky	Aadhira	Gowtham	Ragul
Brief Explanation of the OS	Architecture of OS	Process Control Mechanisms	Scheduling Algorithms	Salient features

Do you have any questions?

ASK US! We hope you learned something new.

"Ubuntu and Debian: Your trusty companions in the vast world of open-source adventures."