

[OPERATING SYSTEM]

[ASSIGNMENT]



Table of Contents

[1ST Paper Title: “A Literature Review on Android -A Mobile Operating system” 3](#_Toc186396285)

[Platform: IRJET 3](#_Toc186396286)

[Link: ANDROID ARCHITECTURE 3](#_Toc186396287)

[Summary of topics: 3](#_Toc186396288)

[1. Process Management in Android OS 3](#_Toc186396289)

[2. File System in Android OS 3](#_Toc186396290)

[3. Memory Management in Android OS 3](#_Toc186396291)

[4. Scheduling in Android OS 4](#_Toc186396292)

[5. Security in Android OS 4](#_Toc186396293)

[Summary of Paper: 5](#_Toc186396294)

[2ND Paper Title: "A Comparative Analysis of Technical Details in Operating Systems" 5](#_Toc186396295)

[Platform: Course Hero 5](#_Toc186396296)

[Link: TECHNICAL DETAILS OF OS 5](#_Toc186396297)

[Summary of Relevant Topics: 5](#_Toc186396298)

[1. Process Management 5](#_Toc186396299)

[2. Memory Management 5](#_Toc186396300)

[3. File System 5](#_Toc186396301)

[4. Scheduling 6](#_Toc186396302)

[5. Security 6](#_Toc186396303)

[Summary of the Paper 6](#_Toc186396304)

[Comparison of the OS concepts 6](#_Toc186396305)

[Creative Analogy and Explanation: 7](#_Toc186396306)

[Insights and Personal Observations on OS Differences: 8](#_Toc186396307)

[Conclusion: 10](#_Toc186396308)

# 1ST Paper Title: “A Literature Review on Android -A Mobile Operating system”

# Platform: IRJET

# Link: [ANDROID ARCHITECTURE](https://www.researchgate.net/publication/354576500_A_Literature_Review_on_Android_-A_Mobile_Operating_system)

# Summary of topics:

## Process Management in Android OS

The **Linux Kernel Layer** in Android is responsible for managing processes. It ensures efficient multitasking and resource allocation by handling process creation, execution, and termination. Each application runs in its own **Dalvik Virtual Machine (DVM)** instance, providing isolation and enabling multiple applications to run simultaneously without interfering with each other. This ensures performance and stability, especially on devices with limited resources.

The Android system uses priority-based process scheduling, where foreground tasks (e.g., the currently active app) are given higher priority over background tasks. Additionally, the **Low Memory Killer** mechanism terminates non-essential background processes when the system is under memory pressure.

## File System in Android OS

The **Linux Kernel Layer** provides support for various file systems, such as **ext3**, **ext4**, and **FAT**, enabling efficient data storage and retrieval. Key components include:

* **SQLite Database**: A lightweight relational database available to all applications, located within the **Native Library Layer**. It provides structured storage for app data.
* **Content Providers**: These are part of the **Application Framework Layer**, allowing secure and controlled sharing of data between applications.
* File access permissions ensure that apps can only interact with files within their designated storage unless explicitly shared.

## Memory Management in Android OS

Android leverages the Linux kernel’s memory management capabilities for efficient RAM utilization. Key features include:

* **Virtual Memory Management**: Applications run within isolated DVM instances, which optimize memory allocation and prevent memory leaks.
* **Garbage Collection**: Managed by the DVM, garbage collection automatically frees unused memory, reducing the risk of crashes and ensuring smoother performance.
* **Low Memory Handling**: When the system runs out of memory, it terminates background processes to prioritize active and high-priority tasks.

The **Just-in-Time (JIT) Compiler** introduced in Android 2.2 further enhances memory efficiency by compiling frequently used code during runtime.

## Scheduling in Android OS

Android uses the Linux kernel's **Completely Fair Scheduler (CFS)** to manage task execution. This ensures:

* **Efficient CPU Allocation**: Foreground processes are prioritized to maintain responsive user interactions.
* **Job Scheduling Framework**: Tasks like background updates or delayed notifications are handled by Android’s job scheduling system, which optimizes battery usage and resource management.
* **Real-Time Scheduling**: Android ensures that time-critical tasks, such as user interface updates and system notifications, are executed without delay.

## Security in Android OS

Android employs a multi-layered security framework, with features at both the system and application levels:

* **Sandboxing**: Each app runs in an isolated environment (a DVM instance), ensuring that its data and processes are inaccessible to other apps.
* **Permissions System**: Users must explicitly grant access to sensitive resources like the camera, location, or contacts.
* **SELinux (Security-Enhanced Linux)**: Incorporated in the **Linux Kernel Layer**, SELinux enforces strict access control policies to prevent unauthorized access.
* **File Encryption**: Android ensures that all sensitive user data is encrypted, protecting it from unauthorized access or tampering.
* **Secure App Deployment**: Applications are distributed in .apk format, which undergoes signature verification to ensure integrity and authenticity.

# Summary of Paper:

The Linux Kernel in Android handles process and memory management, ensuring efficient multitasking with mechanisms like the Low Memory Killer and Dalvik Virtual Machine (DVM) for app isolation. File systems such as ext4 support data storage, with SQLite and Content Providers enabling structured and secure data sharing. Scheduling uses the Completely Fair Scheduler (CFS) for resource optimization and real-time responsiveness. Security is ensured through sandboxing, SELinux policies, encryption, and a robust permissions system, protecting apps and user data.

# **2ND Paper Title:** "A Comparative Analysis of Technical Details in Operating Systems"

# **Platform:** Course Hero

# Link: [TECHNICAL DETAILS OF OS](https://www.coursehero.com/file/228488683/A-Comparative-Analysis-of-Technical-Details-in-Operating-Systemsdocx/)

# Summary of Relevant Topics:

## **Process Management**

The paper examines how different operating systems, including macOS, handle process creation, execution, and termination. It discusses the strategies employed to manage multiple processes efficiently, ensuring optimal CPU utilization and system responsiveness.

## **Memory Management**

It explores the memory allocation techniques used by macOS, such as dynamic allocation and virtual memory management, to ensure efficient utilization of system memory and prevent issues like fragmentation.

## **File System**

The study analyzes the file system architecture of macOS, detailing how it organizes, stores, and retrieves files. It highlights the hierarchical structure and the use of metadata to enhance file management and access speed.

## **Scheduling**

The paper discusses the CPU scheduling algorithms implemented in macOS to manage process execution order, aiming to optimize performance and ensure fair resource distribution among processes.

## **Security**

It reviews the security mechanisms integrated into macOS, including user authentication, access controls, and encryption methods, designed to protect the system against unauthorized access and potential threats.

# ****Summary of the Paper****

The paper provides a **comparative analysis** of modern operating systems, including **macOS**, focusing on their internal mechanisms. It explores fundamental concepts like **process management, memory management, file systems, scheduling, and security** to highlight differences and similarities between macOS and other operating systems such as Windows and Linux.

Comparison of the OS concepts

|  |  |  |
| --- | --- | --- |
| **OS CONCEPT** | **ANDROID** | **IOS** |
| **PROCESS MANAGEMENT** | * Uses Linux kernel for process management. * Processes created using fork(). * IPC via Binder. | * Uses Mach and BSD components. * Priority-based scheduling. * IPC via Mach messaging and higher-level APIs. |
| **MEMORY MANAGEMENT** | * Uses ART (Android Runtime) for memory management. * Garbage collection (older versions: Dalvik VM). * Linux kernel provides virtual memory. | * Uses ARC (Automatic Reference Counting). * Virtual memory managed by Mach kernel. * Strong memory protection. |
| **FILE SYSTEM** | * Uses ext4 filesystem (via Linux VFS). * Partitions: /system, /data, /cache. | * Uses APFS (Apple File System). * Optimized for mobile devices. * Includes encryption and snapshot capabilities. |
| **SCHEDULING** | * Uses Linux kernel’s Completely Fair Scheduler (CFS). * Not real-time,   optimized for mobile. | * Priority-based scheduler. * Real-time processing for UI tasks. |
| **SECURITY** | * Permission-based model. * Full-disk encryption. * Open-source nature can introduce vulnerabilities. | * Sandboxing of apps. * AES-256 encryption. * Secure Enclave for sensitive data. |

# Creative Analogy and Explanation:

Imagine each operating system as a bustling city:

* **Android City:** Built on a robust Linux foundation, it allows each app (citizen) to live in its own apartment (process) within high-rise buildings. The city's infrastructure ensures that citizens can communicate efficiently through a central postal system (Binder), and memory lanes are managed to prevent overcrowding, ensuring smooth traffic flow. The city's security is maintained through a permission-based system, where citizens need passes to access certain areas, and the entire city is under surveillance to protect against intrusions.
* **iOS City:** A meticulously planned metropolis where each app resides in a gated community (sandbox) with strict security protocols. The city's management ensures that resources are allocated efficiently, with each community having its own dedicated space (memory). Communication between communities is regulated through secure channels (Mach messaging), and the city's scheduling ensures that all activities occur in a timely and organized manner. Advanced security measures, including facial recognition at entry points, ensure that only authorized individuals have access to specific areas (system resources). Each community (application) operates independently, preventing unauthorized access or interference from others, ensuring stability and reliability within the metropolis.

# Insights and Personal Observations on OS Differences:

1. **Process Management:**
   * **Android:** Android uses the Linux kernel for process management, which creates a stable foundation for multitasking. However, because Android is a mobile OS designed to optimize battery life and resource usage, it limits the complexity of process management compared to desktop systems. The use of the Binder IPC mechanism is efficient but somewhat restrictive for more complex communication tasks.
   * **iOS:** iOS follows a similar approach to Android but with more stringent controls and optimizations, especially around security. The sandboxing of apps is a unique feature in iOS, ensuring apps cannot directly interfere with system-level processes or other apps, which enhances security but can limit the flexibility for developers.

**Personal Insight:** iOS and Android are more streamlined in terms of process management due to the mobile nature of the platforms, while macOS offers more flexibility, especially for power users and those who require robust multitasking. The limitations of mobile OS process management, while optimizing for performance and battery life, can sometimes feel restrictive in terms of functionality and app interaction.

1. **Memory Management:**
   * **Android:** The use of ART (Android Runtime) helps manage memory efficiently, but it also leads to some challenges, particularly with large apps and memory-intensive operations. The need for garbage collection and managing apps that run in the background can sometimes result in inefficient memory usage.
   * **iOS:** iOS's memory management is more controlled thanks to ARC (Automatic Reference Counting), which is highly effective in managing resources without overburdening the system with garbage collection. However, this also means developers must be very careful with object references, as improper handling can lead to memory leaks.

**Personal Insight:** While iOS and Android prioritize resource conservation for mobile environments, macOS is built for performance, handling more extensive memory needs. However, macOS’s memory management might seem too complex and over-engineered when compared to the simplicity of mobile platforms.

1. **File Systems:**
   * **Android:** The use of the ext4 file system provides a stable, mature solution for Android, but the lack of sophisticated features like snapshots or cloning limits the OS's ability to handle more complex use cases. The need for partitioning (e.g., /system, /data) reflects the nature of Android as a mobile system with a strong focus on user data segregation and security.
   * **iOS:** APFS is a game-changer for iOS, offering advanced features such as encryption, snapshots, and high efficiency for mobile storage. This is ideal for mobile users who need fast, secure file management without the overhead found in traditional desktop systems.

**Personal Insight:** The contrast in file systems highlights the core differences in how these OSes cater to their user base. iOS and Android focus on efficient, lightweight file management for mobile use, while macOS provides a more robust and feature-rich solution for desktop and professional environments.

1. **Scheduling:**
   * **Android:** The Linux kernel's CFS (Completely Fair Scheduler) handles CPU scheduling, which works well for mobile devices. However, Android's scheduler is not as fine-tuned for real-time tasks, as the focus is more on energy efficiency and responsiveness for general usage.
   * **iOS:** iOS features a similar priority-based scheduler, but with more stringent real-time capabilities, ensuring smooth user interactions. The OS is designed to prioritize system responsiveness, especially for UI tasks and real-time communications.

**Personal Insight:** The difference in scheduling priorities between these systems shows how each OS is tailored for its user base. Android and iOS focus on responsiveness and energy efficiency, while macOS prioritizes system throughput and multi-tasking capabilities, especially in professional settings where heavy computing tasks are common.

1. **Security:**
   * **Android:** While Android offers strong security with its permission-based model, its open-source nature can sometimes lead to security vulnerabilities, especially with third-party apps. The Google Play Store's vetting process has improved, but the ecosystem remains prone to malware and untrusted apps.
   * **iOS:** iOS's security model is among the strongest in the mobile world, with rigorous app vetting, sandboxing, and secure enclave hardware for sensitive data storage. Apple’s strict control over the app ecosystem ensures fewer vulnerabilities, but the trade-off is a less open system.

**Personal Insight:** iOS stands out for its uncompromising security due to its closed ecosystem, whereas Android's security is less stringent but more flexible. macOS, while secure, presents a middle ground that allows more flexibility but requires greater user vigilance to maintain security.

# Conclusion:

These operating systems are designed with different priorities in mind: Android and iOS are optimized for mobile environments, focusing on battery life, security, and resource efficiency, while macOS is built for performance and flexibility in a desktop environment. Understanding these differences can help users and developers choose the right platform for their needs.