

Operating Systems

**Assignment #03**

Name: Hira Usman

Roll.no: 221406

Class: BS-CS-V-C

Date:26th December,2024

# Report

# Summary of Research Papers:

### ****Paper 1: A Literature Review on Android - A Mobile Operating System****

**Introduction**:  
This literature review explores the Android Operating System (OS), which is built on the Linux kernel and developed by Google. Android has become one of the most widely used mobile platforms globally.

**Key Insights**:

* **Architecture**: Android consists of four key layers: Linux Kernel, Native Libraries, Application Framework, and Applications. It’s designed to provide flexibility and performance, particularly in mobile environments.
* **Dalvik Virtual Machine (DVM)**: The DVM is designed to optimize Android for low-memory devices, employing techniques like Just-In-Time (JIT) compilation.
* **Version Evolution**: Android has evolved from version 1.0 to 11, introducing significant updates, such as security enhancements and UI improvements. Android moved from dessert-based naming to numerical versions post-Android 9.
* **Advantages & Disadvantages**: Android is open-source and highly customizable but faces security vulnerabilities and requires specialized tools for efficient development.
* **Critical Observations**: Android’s open-source nature has allowed rapid innovation, but its reliance on the Dalvik VM introduces challenges. The OS continues to adapt to emerging trends while facing backward compatibility issues.

**Conclusion**:  
Android OS has evolved significantly, offering scalability and versatility. However, challenges in security and resource optimization remain.

### ****Paper 2: A Comparative Study of Operating Systems****

**Introduction**:  
This research compares six major operating systems—Windows, UNIX, Linux, Mac, Android, and iOS—focusing on their strengths, weaknesses, and architecture.

**Key Insights**:

* **Architecture**: The study highlights differences in kernel architectures: Windows and Android use hybrid kernels, while UNIX, Linux, and Mac use monolithic kernels. iOS uses a modified XNU kernel.
* **Usability**: Windows and Mac are known for their user-friendliness, whereas UNIX and Linux cater more to advanced users. Android is user-friendly but varies by device.
* **Security**: iOS and UNIX are more secure compared to Android, which faces higher malware risks. Linux offers customization, but lacks extensive vendor support.
* **Compatibility**: Windows and Android offer broad compatibility with many devices, while macOS and iOS are more restrictive, being tied to Apple hardware.
* **Strengths & Weaknesses**:
  + **Windows**: Broad compatibility, but prone to malware.
  + **UNIX**: Stable but complex to set up.
  + **Linux**: Open-source but has a steep learning curve.
  + **Mac**: Excellent integration but expensive and limited compatibility.
  + **Android**: Flexible and open-source but vulnerable to security threats.
  + **iOS**: Highly secure and optimized, but expensive and closed.

**Conclusion**:  
The comparative study offers insights into OS selection based on factors like security, usability, performance, and cost, with a focus on cross-platform compatibility and security concerns

### ****Paper 3: A Comprehensive Study of Kernel (Issues and Concepts) in Different Operating Systems****

**Introduction**:  
This study provides an in-depth analysis of the kernel in various operating systems (Windows, Linux, iOS, Android, and macOS), focusing on the kernel's architecture, key features, issues, and the influence of emerging technologies like IoT and cloud computing.

**Key Insights**:

* **Kernel Fundamentals**: The kernel acts as the interface between hardware and software, providing essential services like process scheduling, memory management, and device interaction.
  + **Microkernel**: Minimalistic, delegating services to user space (e.g., Linux).
  + **Monolithic Kernel**: Incorporates most services in kernel space (e.g., UNIX, Linux).
  + **Hybrid Kernel**: Combines aspects of both microkernel and monolithic (e.g., Windows, macOS).
* **Operating Systems and Kernel Features**:
  + **Windows**: Hybrid kernel architecture, user-friendliness, and compatibility.
  + **Linux**: Monolithic kernel, highly customizable, ideal for servers.
  + **iOS/macOS**: Hybrid kernel with security and integration for Apple devices.
  + **Android**: Based on the Linux kernel, optimized for mobile devices with open-source flexibility.
* **Emerging Technologies**: IoT and cloud computing require kernel adaptations for virtualization, security, resource allocation, and real-time processing. Security, scalability, and resource sharing are ongoing challenges in kernel design.
* **Kernel Issues and Solutions**:
  + **Security**: Rootkits and privilege escalation remain major vulnerabilities.
  + **Performance**: Task scheduling and memory management bottlenecks are common.
  + **Compatibility**: Ensuring seamless operation across diverse environments is challenging.
  + **Proposed Solutions**: Use of hardware-assisted virtualization, adaptive scheduling algorithms, and kernel extensions like COMEX for memory management.

**Conclusion**:  
The kernel plays a critical role in the functioning of modern operating systems, and its design must evolve to address emerging technological trends and increasing demands for scalability, security, and efficiency.

# 2. Comparison of OS concepts:

## **1. Process Management**

#### ****Android OS**** :

* **Process Creation**: In Android, processes are created when an app or service is launched. Each app runs in its own isolated process, ensuring security and stability. The system uses a **fork-exec** model for process creation, where an existing process (like a system process) spawns new processes (for apps or services).
* **Process Scheduling**: Android uses the **Linux Kernel** for process scheduling. It supports **preemptive multitasking**, allowing the kernel to manage the execution of multiple processes by assigning CPU time to processes based on their priority and resource requirements.
* **Multitasking**: Android is optimized for multitasking, especially on mobile devices with limited resources. Active apps and services are allowed to run concurrently, while background apps are paused or killed to conserve battery life and memory.
* **Inter-process Communication (IPC)**: Android uses **Binder** for IPC, which allows processes to communicate with each other in a secure and efficient manner. Binder facilitates communication between applications and system services, which is crucial for Android's multi-application environment.

#### ****macOS**** :

* **Process Creation**: macOS follows a **UNIX-like** process creation model. The **XNU Kernel** (Hybrid of Mach and BSD) creates processes when applications are launched. Processes are isolated, but macOS allows for multitasking across multiple applications.
* **Process Scheduling**: macOS, like Android, uses **preemptive multitasking**. The **XNU Kernel** is responsible for scheduling processes based on priority, with the scheduler deciding which process runs on the CPU at any given time.
* **Multitasking**: macOS efficiently supports multitasking, allowing users to run multiple applications simultaneously. It manages multiple tasks using cooperative scheduling, which works well on powerful hardware.
* **Inter-process Communication (IPC)**: macOS uses several IPC mechanisms, such as **Mach ports** (from the Mach microkernel) and **BSD sockets**, to allow communication between processes. This enables seamless interactions between system processes and user applications.

## **2. Memory Management**

#### ****Android OS**** :

* **Memory Allocation and Deallocation**: Android's memory management is handled by the **Linux Kernel** and its **Android Runtime (ART)**. The kernel uses **paging** and **virtual memory** to allocate memory dynamically. Apps are granted specific memory regions, and the system manages memory using a **garbage collection** mechanism to reclaim memory that is no longer in use.
* **Virtual Memory**: Android uses **virtual memory** to give each app the illusion that it has its own memory space, even if the system's physical memory is limited. This is achieved through **paging** and **swapping**, where data not actively used is written to disk.
* **Caching**: Android uses caching to improve performance, including **memory cache** for quicker access to data that is frequently accessed (e.g., images, web content).
* **Memory Protection**: Android ensures memory protection using **sandboxes** for each app, preventing one app from accessing another’s memory space.

#### ****macOS**** :

* **Memory Allocation and Deallocation**: macOS uses **XNU Kernel**'s advanced memory management techniques, including **paging**, **virtual memory**, and **memory compression** to ensure efficient memory usage. The system manages memory dynamically, allocating and deallocating memory as needed.
* **Virtual Memory**: Like Android, macOS supports **virtual memory** to ensure that each application has its own virtual address space, even when physical memory is fully utilized. It uses **paging** and **swapping** to ensure smooth operation when the memory demand exceeds physical RAM.
* **Caching**: macOS implements **high-performance caching** techniques to reduce the time it takes to retrieve data. The **App Nap** feature also helps optimize memory usage by putting idle apps to sleep and reducing their resource consumption.
* **Memory Protection**: macOS uses various memory protection techniques to prevent unauthorized memory access. The **XNU kernel** includes features like **System Integrity Protection (SIP)** to safeguard critical system files and processes from modification.

## **3. File System**

#### ****Android OS**** :

* **File Storage and Access**: Android uses the **ext4** file system, which is optimized for flash storage. Android stores files on internal storage or external SD cards. Apps and system files are isolated from each other using sandboxing.
* **File Organization**: Android organizes files hierarchically, with **/data** for app data, **/system** for the OS, and **/cache** for temporary files. External storage (SD cards) is used for media and app data.
* **Differences in File System Structures**: Android primarily uses **ext4**, which is common in Linux-based systems. The ext4 file system supports large files, efficient storage management, and journal-based data integrity.

#### ****macOS**** :

* **File Storage and Access**: macOS uses the **APFS (Apple File System)** for its internal drives, which is designed for modern storage devices (e.g., SSDs). It also supports **HFS+** for legacy storage devices.
* **File Organization**: macOS organizes files similarly to Unix-based systems, with **/System**, **/Users**, and **/Applications** directories. APFS introduces features like **cloning** (efficient file copying) and **snapshots** (creating system backups).
* **Differences in File System Structures**: **APFS** is optimized for solid-state drives (SSDs) and provides improved security, encryption, and performance compared to **HFS+**. **ext4** in Android, on the other hand, is designed for flash storage, offering high performance and stability.

## **4. Security**

#### ****Android OS**** :

* **Security Mechanisms**: Android employs various security features, such as **App Sandboxing** to isolate apps from each other, **SELinux** for mandatory access control, and **encryption** to secure sensitive data.
* **Permissions**: Android uses **permissions** to control access to sensitive data (e.g., camera, contacts, location). Apps request permissions at runtime, and users must explicitly grant or deny them.
* **Authentication**: Android supports various authentication methods, including **fingerprint recognition**, **PIN**, and **face recognition**, as well as **encryption** for securing user data.

#### ****macOS**** :

* **Security Mechanisms**: macOS includes robust security features, such as **FileVault** (full disk encryption), **Gatekeeper** (ensuring only trusted apps can be installed), and **System Integrity Protection (SIP)** to prevent unauthorized modifications to system files.
* **Permissions**: macOS uses **permissions** based on **Unix file access control**. The system employs **sandboxing** for apps to restrict their access to critical resources and files.
* **Authentication**: macOS supports various authentication mechanisms, including **Touch ID**, **Apple ID**, **FileVault**, and **two-factor authentication** for secure login.

## **5. Scheduling**

#### ****Android OS**** :

* **CPU Scheduling Algorithms**: Android uses the **Linux Kernel's** scheduler, which implements **CFS (Completely Fair Scheduler)** for general-purpose scheduling. This ensures that CPU time is allocated fairly across all processes and threads.
* **Real-Time Processing**: Android does not prioritize real-time tasks, but the kernel’s scheduler ensures that time-sensitive apps (like media players) get their required resources.
* **Multiple Users/Processes**: Android manages multitasking by handling **foreground and background apps**. It also uses **process priority** to determine which app/process should be allocated resources.

#### ****macOS**** :

* **CPU Scheduling Algorithms**: macOS uses the **XNU Kernel**, which employs a **multi-level feedback queue** for process scheduling. It prioritizes tasks based on their CPU usage and responsiveness.
* **Real-Time Processing**: macOS supports real-time processing for applications that require high responsiveness, especially in creative fields (e.g., video editing, music production).
* **Multiple Users/Processes**: macOS is designed for multitasking, with efficient handling of multiple processes and user interactions. It provides **preemptive multitasking** with a focus on maximizing user experience.
* Handles core OS functions, like process scheduling and memory management.
* Includes libraries like OpenGL for graphics and ART for running applications.
* Provides tools like Activity Manager, Content Providers, and Resource Manager for app development.

# OS Architecture Diagram Flow for Android

* macOS directly optimizes its OS for Apple hardware.
* Combines Mach for multitasking and BSD for Unix functionalities.
* Metal for graphics, CoreML for AI, and AppKit for UI development.

# OS Architecture Diagram Flow for MacOS

* macOS-native apps like Safari and Photos interact seamlessly with hardware and software.

# 3. Personal Insights and Observations on different OS

* Flexibility vs. Control: Android’s open-source nature allows for incredible flexibility, enabling developers and users to customize their experience. However, this also means that it can sometimes feel like a chaotic factory floor where things can get messy and unorganized. In contrast, macOS’s more controlled environment is like a polished, high-end factory where everything is carefully managed, resulting in a smoother experience but at the cost of less customization.
* Security: While both systems emphasize security, macOS tends to be more stringent. It’s like a factory with security checkpoints at every door and employee monitored every minute. Android, being open-source, is more like a factory where workers can come and go as they please, but security measures are still in place—just with more potential for breaches.
* Efficiency and Performance: macOS is more streamlined, offering optimized performance with its tailored integration of hardware and software. Android, while flexible, sometimes struggles with resource optimization due to its diverse range of devices and configurations. It's like the difference between a factory that customizes its machinery for specific tasks versus one that must handle all types of machines.

In essence, **Android** is like a dynamic, ever-evolving factory built for versatility, while **macOS** is a highly organized, premium factory that excels in controlled environments and high security. Each has its own strengths and weaknesses depending on the type of product you are producing.