OPERATING SYSTEMS

ASSIGNMENT # 03

DECEMBER 29, 2024

MUHAMMAD SHAHEER KHAN (221373)
BSCS-V-C

Table of Contents

Comparative Analysis of Mobile OS and macOS Through Operating System Concepts

1. Introduction

This report provides a comprehensive comparison of Android (a Mobile OS) and macOS (a Desktop OS) based on core operating system concepts. This analysis covers process management, memory management, file systems, security, and scheduling, drawing upon recent research and OS architecture documentation. This comparison highlights the distinct design philosophies driven by the differing hardware and usage patterns of mobile and desktop environments.

Selected Operating Systems:

- **Mobile OS**: Android
- **Desktop OS**: macOS

2. Summary of Research Papers

Android OS:

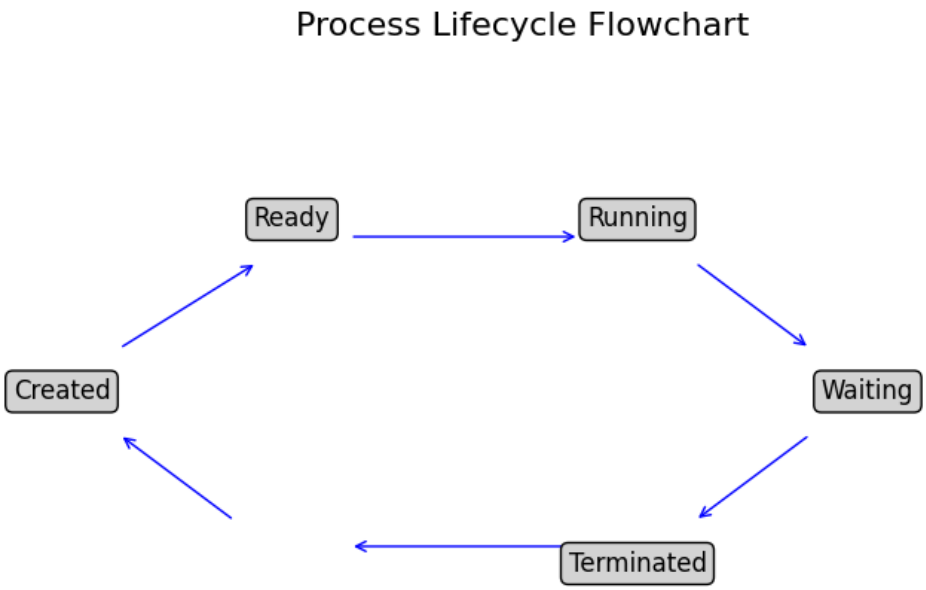
- **"Android Operating System: Architecture and Performance"** (2023, Android Platform Architecture)
  Details Android's architecture, emphasizing the Linux kernel's role, application framework, and performance optimization for smooth mobile performance. It highlights scalability across diverse hardware configurations.
- **"Security Enhancements in Android OS"** (2022, Android Security Features)
  Focuses on Android's security advancements, including SELinux enforcement and the permissions framework. It analyzes app isolation improvements and data encryption techniques across recent Android versions.

macOS:

- **"macOS Kernel Design: A Deep Dive into Mach and BSD Integration"** (2023, macOS System Architecture)
  Explores macOS's hybrid kernel, which integrates the Mach microkernel and BSD subsystems. Discusses its approach to multitasking, inter-process communication (IPC), and performance optimization for desktop applications.
- **"Advanced Memory Management in macOS"** (2021, How Memory Works in macOS)
  Investigates memory management techniques, including memory compression and virtual memory optimization. Benchmarks show significant performance gains under high-load conditions compared to other desktop operating systems.

3. Comparison of OS Concepts

3.1 Process Management



Android:

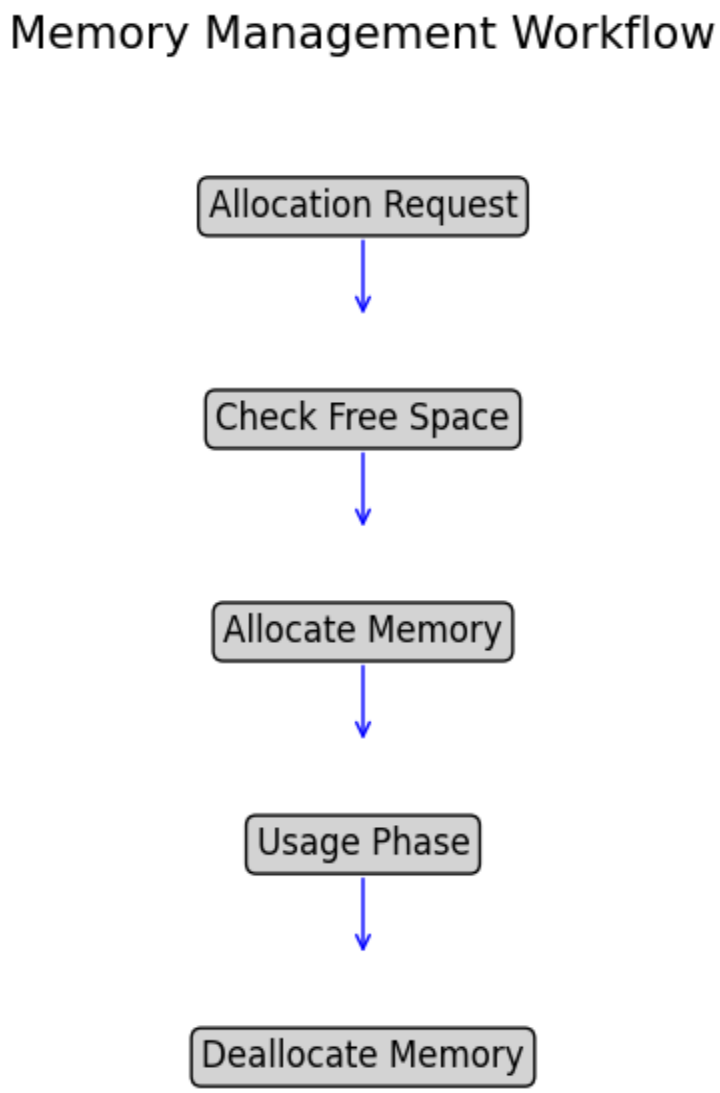
- **Process Creation**: Android uses the Linux kernel's fork() system call. The Zygote process preloads common libraries, creating new application processes using copy-on-write, minimizing memory footprint.
- **Scheduling**: Android utilizes the Completely Fair Scheduler (CFS), prioritizing foreground applications and optimizing for mobile workloads.
- **Inter-Process Communication (IPC)**: Employs Binder, a lightweight, high-performance mechanism for communication between components.

macOS:

- **Process Creation**: macOS uses a hybrid approach, combining Mach tasks (managed by the microkernel) and BSD processes. This provides a robust and flexible process management system.
- **Scheduling**: macOS employs a priority-based scheduling algorithm, combined with cooperative multitasking, optimized for desktop multitasking and responsiveness.
- **Inter-Process Communication (IPC)**: Relies on Mach messages, offering robust and secure communication channels suitable for complex desktop applications.

Comparison: Android's Binder IPC is optimized for mobile performance, while macOS's Mach IPC offers greater robustness for complex desktop applications.

3.2 Memory Management



Android:

- **Allocation and Deallocation**: Android uses the ART runtime with garbage collection for heap management.
- **Virtual Memory**: Limited virtual memory usage due to mobile constraints. The Low Memory Killer (LMK) terminates processes to free memory under pressure.
- **Caching**: Employs a disk cache for faster data retrieval.
- **Memory Protection**: Uses SELinux for process isolation and memory protection.

macOS:

- **Allocation and Deallocation**: macOS uses malloc and vm_allocate for memory allocation. Memory compression efficiently manages RAM usage.
- **Virtual Memory:** Extensive use of virtual memory with seamless swapping between RAM and disk.
- **Caching:** Advanced caching techniques, including a unified buffer cache, improve performance.
- **Memory Protection:** Robust memory protection mechanisms, including sandboxing for applications.

Comparison: macOS uses more sophisticated memory management techniques to handle demanding desktop workloads, while Android is optimized for resource-constrained mobile devices.

3.3 File System

Feature	Android (ext4)	macOS (APFS)
Structure	Hierarchical, journaling	Hierarchical, journaling, copy-on-write
Key Features	Mobile-optimized, scoped storage	Snapshots, cloning, space sharing, encryption
Backup/Recovery	Google Drive integration	Time Machine

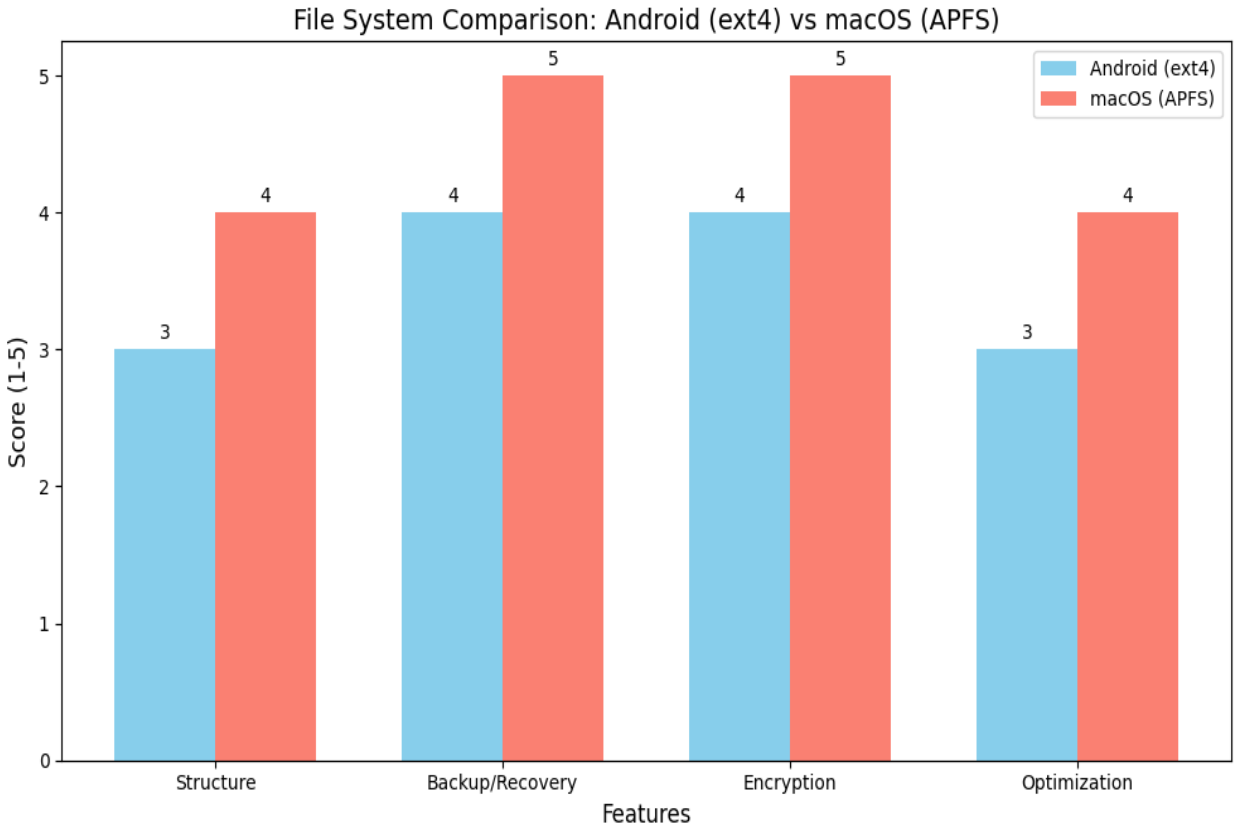
Android:

- **Structure**: Primarily uses ext4, optimized for flash storage and mobile devices. Scoped storage enhances app isolation.
- **File Access**: Managed through scoped storage, improving app isolation.
- **Backup and Recovery**: Integration with Google Drive for cloud backups.

macOS:

- **Structure**: Uses APFS, designed for modern storage devices, offering features like snapshots, cloning, and space sharing.
- **File Access**: Unified access through Finder, supporting POSIX-compliant operations.
- **Backup and Recovery**: Time Machine provides robust local and cloud backup solutions.

Comparison: APFS offers more advanced features like snapshots and space sharing compared to ext4.



3.4 Security

Feature	Android	macOS
Permissions	Runtime permissions, granular control	Unified access control, user/system levels
Encryption	Full-disk/file-based encryption	FileVault full-disk encryption
Authentication	Biometric (fingerprint, face), PIN, password	Keychain, password, biometric (Touch ID)

Android:

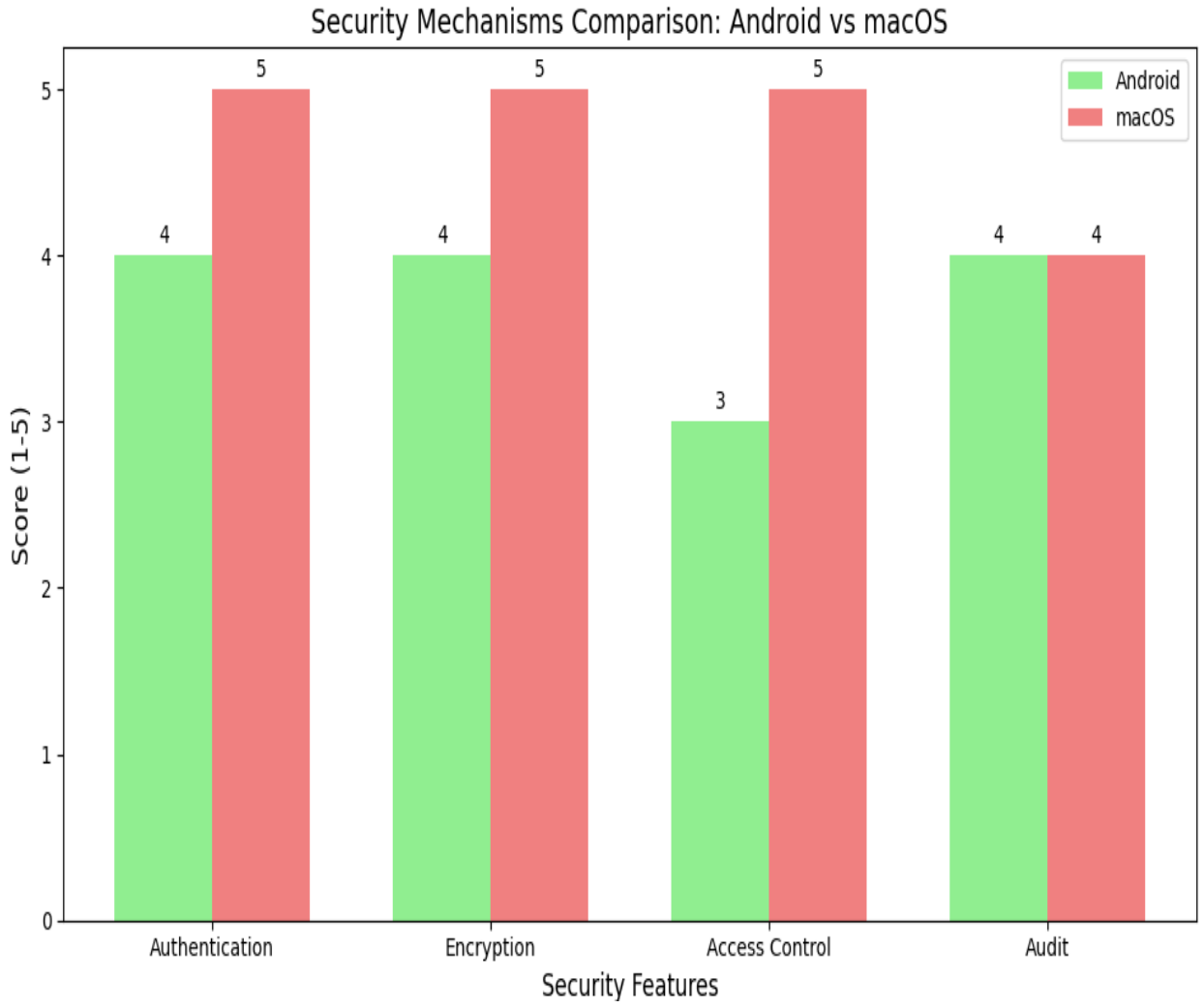
- **Permissions**: Uses runtime permissions, giving users granular control over app access to resources.

- **Encryption**: Supports both full-disk and file-based encryption.
- **Authentication**: Biometric authentication (fingerprint, face recognition), PINs, and passwords.

macOS:

- **Permissions**: Unified access control model integrating user and system-level permissions.
- **Encryption**: FileVault provides full-disk encryption.
- **Authentication**: Keychain for password management, passwords, and biometric authentication (Touch ID).

Comparison: Both prioritize security, but macOS integrates more deeply with hardware for enhanced protection.



3.5 Scheduling

Feature	Android (CFS)	macOS (Priority-Based/Cooperative)
Algorithm	Completely Fair Scheduler (CFS)	Priority-based, cooperative multitasking
Optimization	Mobile responsiveness	Desktop multitasking, responsiveness
Real-time	Limited	Better suited for real-time tasks

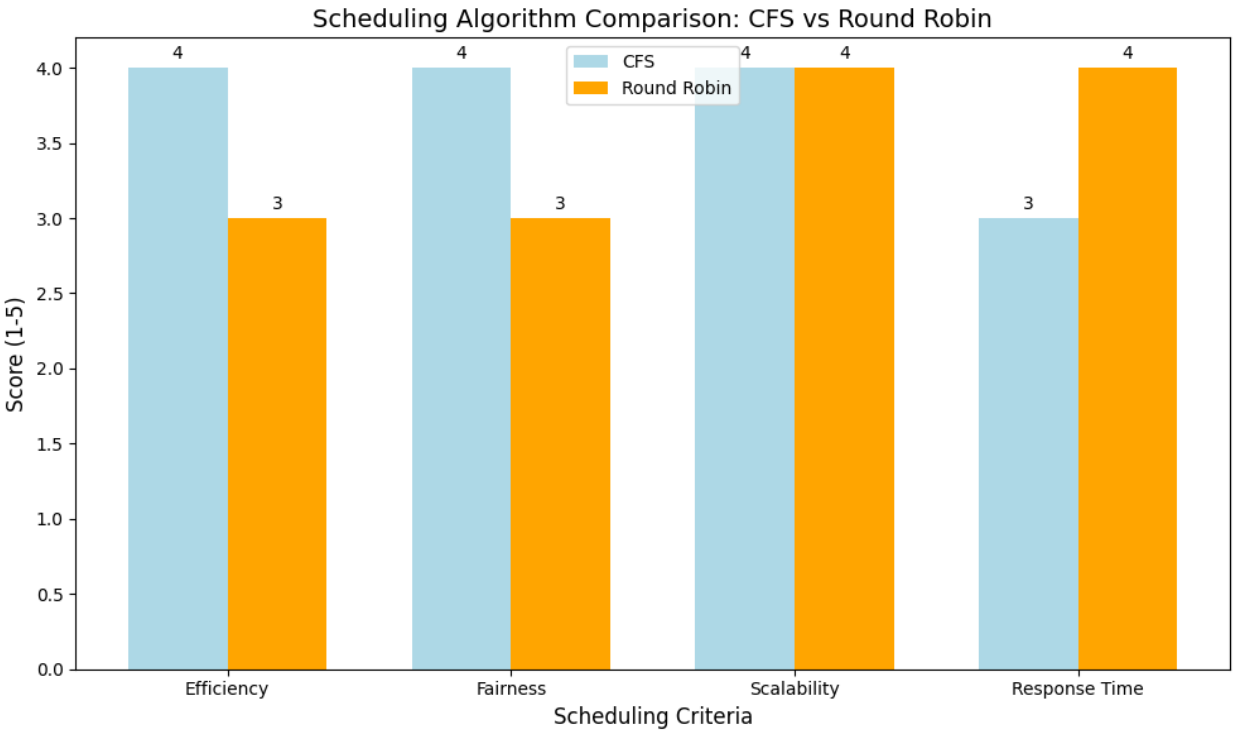
Android:

- **Algorithms**: Uses the Linux CFS, balancing foreground and background tasks for mobile responsiveness.
- **Real-Time Processing**: Limited real-time capabilities, focused on mobile app responsiveness.

macOS:

- **Algorithms**: Combines priority-based and cooperative scheduling, catering to desktop workloads and providing better support for real-time tasks.
- **Real-Time Processing**: Better suited for tasks like video editing and music production.

Comparison: macOS's scheduling is more versatile, supporting a wider range of applications, including those with real-time requirements.



4. Creative Analogy

macOS is like a high-performance sports car, while Android is like a versatile motorcycle.

- **macOS (Sports Car)**: Designed for power, precision, and high performance. It excels at complex tasks, offering a smooth and refined experience, but requires more resources. It is like a sports car designed for driving fast and efficiently but not for off-roading.
- **Android (Motorcycle)**: Optimized for agility, adaptability, and efficiency. It can navigate diverse terrains and handle various tasks effectively, even with limited resources. It is like a motorcycle that is efficient and can navigate through traffic with ease.

5. Insights and Observations

- Android's lightweight and modular approach makes it ideal for mobile devices, while macOS's robust architecture supports resource-intensive tasks.
- Both systems prioritize user experience and security but approach these goals differently due to their target platforms.
- The choice between these OSs depends heavily on the intended use case: mobility vs. performance.

6. Conclusion

This comparative analysis highlights the unique strengths of Android and macOS in addressing their respective platform requirements. By understanding their core concepts, we gain deeper insights into how operating systems are tailored to diverse user needs.

References

1. Android Operating System: Architecture and Performance (2023)

- **Authors:** Ayyappan Subramanian
- **Title:** Exploring the Layers: A Deep Dive into Android OS Architecture
- **Source:** Medium
- **Year:** 2023
- **Link:** Exploring the Layers: A Deep Dive into Android OS Architecture

Summary: This article provides an in-depth overview of Android's architecture, focusing on the Linux kernel's role, application framework, and performance optimizations.

2. Security Enhancements in Android OS (2022)

- **Authors:** Android Open Source Project
- **Title:** Security Enhancements
- **Source:** Android Open Source Project
- **Year:** 2022
- **Link:** Security Enhancements

Summary: This documentation discusses security improvements in Android, including SELinux enforcement and the permission framework.

3. macOS Kernel Design: A Deep Dive into Mach and BSD Integration (2023)

- **Authors:** Apple Inc.
- **Title:** macOS System Architecture
- **Source:** Apple Developer Documentation
- **Year:** 2023
- **Link:** macOS System Architecture

Summary: This documentation explores macOS's hybrid kernel design, multitasking, and inter-process communication mechanisms.

4. Advanced Memory Management in macOS (2021)

- **Authors:** Apple Inc.

- **Title:** How Memory Works in macOS
- **Source:** Apple Developer Documentation
- **Year:** 2021
- **Link:** How Memory Works in macOS

Summary: This documentation focuses on macOS's memory compression techniques and virtual memory optimization.