RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

Department of Master of Computer Applications (MCA)

Mobile Application Development (MCA221IA)

Hand Notes

Unit - Unit-1	**Topic:** Operating System
Editor: USN- 1RV24MCO91 **Student Name**- Sameer Patel	

List of Questions

2 MARKS QUESTIONS (4 Questions)

1. **What is a Mobile Operating System? Give two examples.**
2. **Differentiate between Native and Cross-platform mobile development.**
3. **What are the main components of Android Architecture?**
4. **What is App Lifecycle Management in mobile OS?**

4 MARKS QUESTION (1 Question)

5. **Explain the Android Activity Lifecycle with a diagram.**

6 MARKS QUESTION (1 Question)

6. **Explain Memory Management in Mobile Operating Systems with examples.**

8 MARKS QUESTION (1 Question)

7. **Compare Android and iOS operating systems in terms of architecture, security, and development approach.**

10 MARKS QUESTION (1 Question)

8. **Explain the process management and inter-process communication mechanisms in mobile operating systems with practical examples.**

Q1.What is a Mobile Operating System? Give two examples.

Answer: A Mobile Operating System is system software that manages hardware resources and provides services for mobile applications to run on smartphones, tablets, and other mobile devices.

Examples:

- Android (Google)
- iOS (Apple)

Q2. Differentiate between Native and Cross-platform mobile development.

Answer:

Native Development	Cross-platform Development
Platform-specific languages (Java/Kotlin for Android, Swift for iOS)	Single codebase for multiple platforms
Better performance and device integration	Faster development and cost-effective

Q3. What are the main components of Android Architecture?

Answer: The main components are:

- **Application Layer** - User apps and system apps
- **Application Framework** - APIs and services
- **Android Runtime** - ART/Dalvik VM
- **Linux Kernel** - Hardware abstraction layer

Q4. What is App Lifecycle Management in mobile OS?

Answer: App Lifecycle Management refers to the different states an application goes through from launch to termination, including:

- **Created/Launched** - App starts
- **Active/Running** - App in foreground
- **Paused** - App partially obscured
- **Stopped** - App in background
- **Destroyed** - App terminated

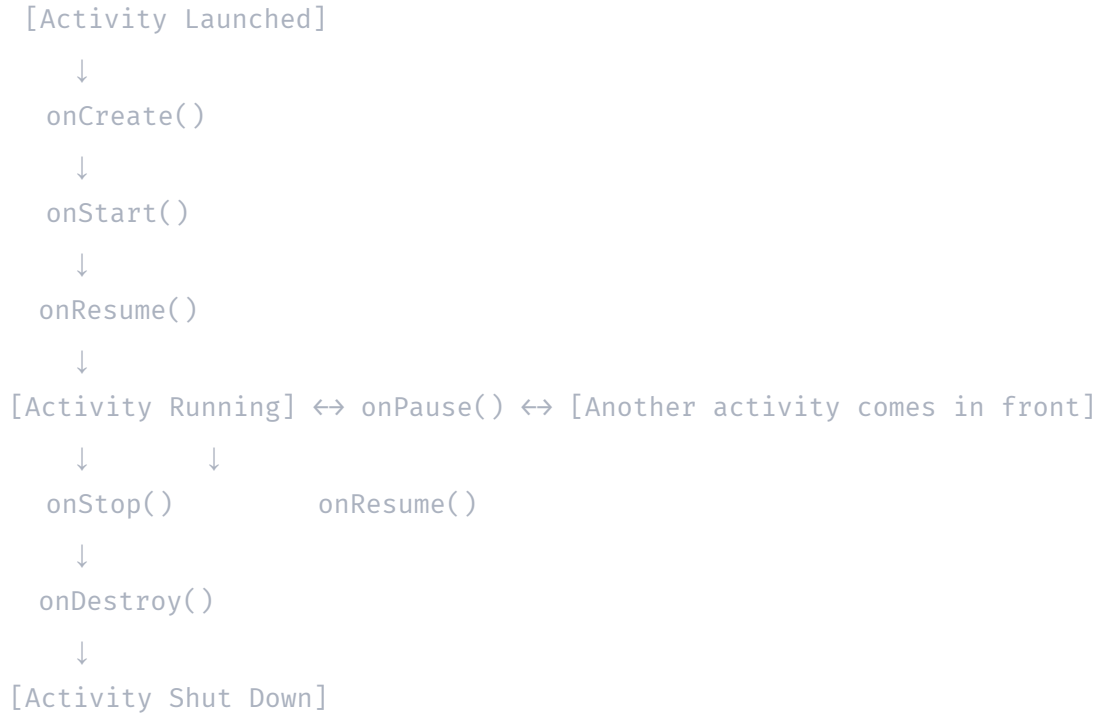
Q5:Explain the Android Activity Lifecycle with a diagram.

Answer: The Android Activity Lifecycle represents the different states an activity goes through during its lifetime.

Lifecycle States:

1. **onCreate()** - Activity is first created
2. **onStart()** - Activity becomes visible to user
3. **onResume()** - Activity gains focus and becomes interactive
4. **onPause()** - Activity loses focus but remains visible
5. **onStop()** - Activity is no longer visible
6. **onDestroy()** - Activity is destroyed and removed from memory

Activity Lifecycle Diagram:



Q6:Explain Memory Management in Mobile Operating Systems with examples.

Answer:

Memory management in mobile OS is crucial due to limited RAM and battery constraints. Mobile OS employ various techniques to optimize memory usage.

Key Memory Management Concepts:

1. Application Memory Allocation:

- Each app runs in its own process with allocated heap memory
- Apps have memory limits (typically 16MB-512MB depending on device)
- Automatic garbage collection removes unused objects

2. Memory Management Techniques:

Technique	Description	Example
Low Memory Killer (LMK)	Kills background apps when memory is low	Android's LMK kills apps based on priority
Memory Compression	Compresses inactive pages in RAM	iOS uses memory compression since iOS 7
App Backgrounding	Suspends apps moved to background	iOS freezes background apps
Garbage Collection	Automatic memory cleanup	ART runtime in Android

3. Memory States in Android:

4. Memory Optimization Strategies:

- Use object pools for frequently created objects
- Implement proper caching mechanisms
- Use appropriate data structures
- Release resources in lifecycle methods
- Monitor memory usage with profiling tools

5. Memory Management Best Practices:

- Avoid memory leaks by properly managing references
- Use WeakReference for views to prevent memory leaks
- Release bitmap resources when not needed
- Close database connections and file streams
- Implement proper cleanup in onDestroy() methods

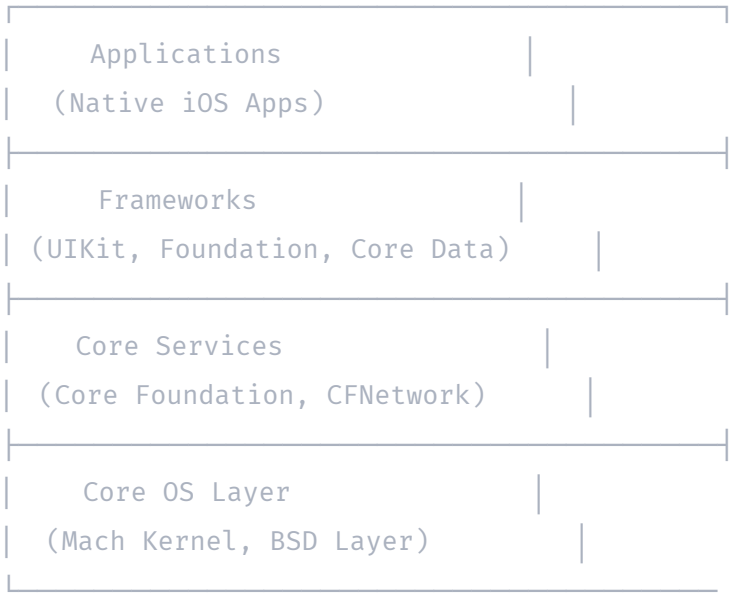
Q7: Compare Android and iOS operating systems in terms of architecture, security, and development approach.

Answer:

1. ARCHITECTURE COMPARISON

Android Architecture:

iOS Architecture:



2. DETAILED COMPARISON TABLE

Aspect	Android	iOS
Kernel	Linux Kernel	Darwin (Mach + BSD)
Runtime	Android Runtime (ART)	Objective-C Runtime
File System	ext4, F2FS	APFS (Apple File System)
Process Management	Linux-based process model	Mach microkernel
Memory Management	Dalvik/ART heap, Linux memory	Reference counting + ARC
UI Framework	Views, Compose	UIKit, SwiftUI
Programming Languages	Java, Kotlin	Swift, Objective-C

3. SECURITY COMPARISON

Security Features Comparison:

Security Feature	Android	iOS
App Sandboxing	Linux UID-based sandbox	Mandatory app sandbox
Code Signing	Optional (Google Play)	Mandatory for all apps
App Store Review	Automated + Manual	Strict manual review
Encryption	File-based encryption	Hardware-level encryption
Biometric Auth	Fingerprint, Face	Touch ID, Face ID
Permission Model	Install-time + Runtime	Runtime permissions
Root Access	Can be rooted	Jailbreaking required

Android Security Features:

- SELinux (Security-Enhanced Linux) for mandatory access control
- Application sandboxing using Linux user IDs
- Google Play Protect for malware scanning
- Runtime permissions (Android 6.0+)
- Hardware-backed keystore

iOS Security Features:

- Hardware-level encryption with Secure Enclave
- App Transport Security (ATS)
- System Integrity Protection (SIP)
- Code signing requirement for all apps
- Automatic security updates

4. DEVELOPMENT APPROACH

Development Tools Comparison:

Tool Category	Android	iOS
IDE	Android Studio (IntelliJ-based)	Xcode
Languages	Java, Kotlin, C/C++	Swift, Objective-C, C/C++
UI Design	XML Layouts, Jetpack Compose	Storyboards, SwiftUI
Testing	JUnit, Espresso, Robolectric	XCTest, XCUITest
Distribution	Google Play Store, Side-loading	App Store (Primary)
Debugging	Android Debug Bridge (ADB)	Instruments, LLDB

Development Philosophy:

- **Android**: Open-source, flexible, multiple device support
- **iOS**: Closed ecosystem, consistency, premium user experience

5. PERFORMANCE CHARACTERISTICS

Performance Comparison:

Aspect	Android	iOS
Hardware Diversity	Wide range of devices	Limited, controlled hardware
Optimization	Generic optimization	Hardware-specific optimization
Memory Usage	Higher due to Java/Kotlin overhead	More efficient native code
Battery Life	Varies significantly by device	Generally consistent and optimized
App Launch Time	Can be slower due to runtime compilation	Faster due to ahead-of-time compilation

Q8: Explain the process management and inter-process communication mechanisms in mobile operating systems with practical examples.

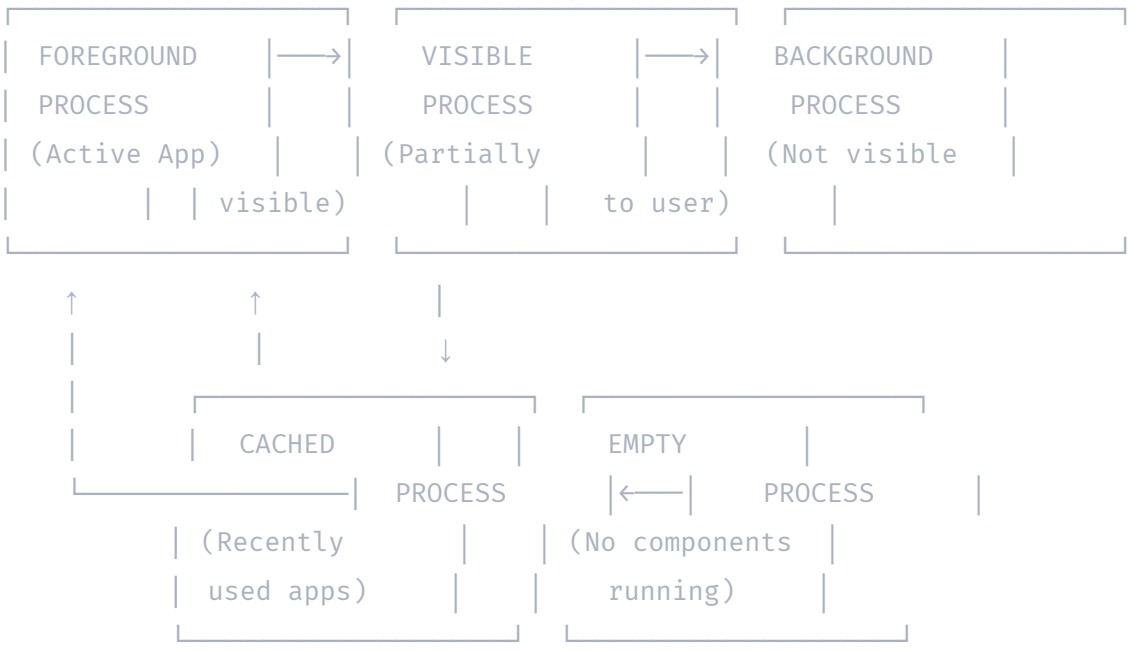
Answer:

PROCESS MANAGEMENT IN MOBILE OPERATING SYSTEMS

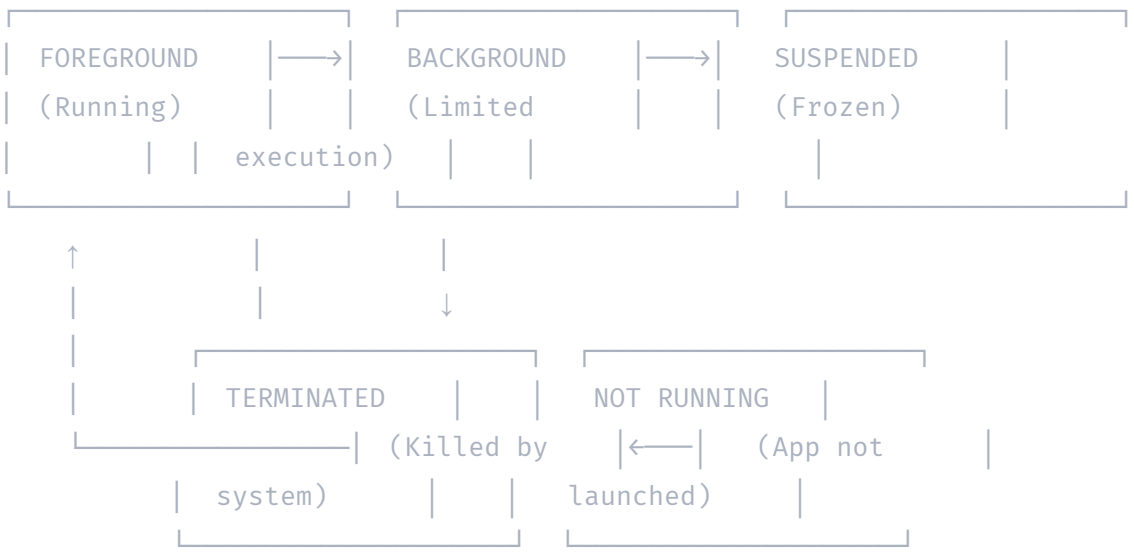
1. PROCESS LIFECYCLE AND STATES

Mobile operating systems manage processes differently from desktop systems due to resource constraints and power management requirements.

Android Process States:



iOS Process States:



2. PROCESS PRIORITY LEVELS

Android Process Priority:

Priority Level	Description	OOM Score
Foreground	Active app user is interacting with	0
Visible	App visible but not in foreground	100
Service	App running background service	200
Background	App in background with activities	400
Empty	Process with no active components	1000

iOS Quality of Service (QoS) Classes:

QoS Class	Priority	Use Case
User Interactive	Highest	UI updates, animations
User Initiated	High	User-requested tasks
Default	Medium	General tasks
Utility	Low	Long-running tasks
Background	Lowest	Maintenance tasks

3. MEMORY MANAGEMENT IN PROCESSES

Android Memory Management:

- **Low Memory Killer (LMK)**: Kills processes based on priority when memory is low
- **Out-of-Memory (OOM) Killer**: Last resort when system is critically low on memory
- **Zygote Process**: Pre-forked process that spawns new app processes
- **Application Memory Limits**: Each app has heap size limits (varies by device)

iOS Memory Management:

- **Automatic Reference Counting (ARC)**: Automatic memory management
- **Memory Pressure Warnings**: System notifies apps to free memory
- **App Backgrounding**: Apps are suspended when moved to background
- **Jetsam**: iOS equivalent of Android's LMK

4. INTER-PROCESS COMMUNICATION (IPC) MECHANISMS

A. Android IPC Mechanisms:

1. Binder IPC:

- Primary IPC mechanism in Android
- Based on OpenBinder, optimized for mobile devices
- Provides security through UID/PID checking
- Supports synchronous and asynchronous communication

Binder Architecture:

2. Intent-based Communication:

- Explicit Intents: Direct component targeting
- Implicit Intents: System resolves appropriate component
- Broadcast Intents: One-to-many communication

Intent Communication Types:

Intent Type	Description	Example Use Case
Explicit	Direct component targeting	Start specific activity
Implicit	System resolves component	Share content
Broadcast	System-wide messages	Battery low notification

3. Content Providers:

- Standardized interface for data sharing
- URI-based access to structured data
- Built-in permission system
- Support for CRUD operations

B. iOS IPC Mechanisms:

1. XPC (Cross Process Communication):

- Apple's IPC framework
- Based on libdispatch and Mach messages
- Provides security through entitlements
- Automatic process lifecycle management

2. Mach Messages:

- Low-level IPC mechanism
- Kernel-level message passing
- Foundation for higher-level IPC
- Port-based communication

3. App Extensions:

- Standardized way to extend app functionality
- Runs in separate process
- Limited lifecycle and capabilities
- Communication through NSExtensionContext

5. PRACTICAL EXAMPLES OF PROCESS COMMUNICATION

Android IPC Example - AIDL Interface:

```
Service Definition:
- MyService extends Service
- Implements IMyService.Stub
- Returns IBinder in onBind()

Client Usage:
- Binds to service using Intent
- Receives IBinder in ServiceConnection
- Converts to interface using Stub.asInterface()
- Makes remote method calls
```

Android Content Provider Example:

```
URI Structure: content://com.example.provider/table/id
Operations: query(), insert(), update(), delete()
Permissions: Read/Write permissions in manifest
Data Types: Cursor-based data access
```

iOS XPC Service Example:

```
Service Configuration:
- XPC service bundle in app package
- Service Info.plist configuration
- Protocol definition for interface

Client Communication:
- NSXPCConnection establishment
- Remote object proxy creation
- Asynchronous method invocation
```

6. PROCESS SCHEDULING

Android Process Scheduling:

- Based on Linux Completely Fair Scheduler (CFS)
- Real-time scheduling for system processes
- Priority inheritance to prevent priority inversion
- CPU affinity for multi-core optimization

iOS Process Scheduling:

- Mach-based scheduling with multiple policies
- QoS-aware scheduling
- Automatic thread QoS propagation
- Energy-aware scheduling

7. SECURITY IN PROCESS MANAGEMENT

Process Isolation Mechanisms:

Security Feature	Android	iOS
Process Sandbox	Linux UID separation	Mandatory sandbox
Permission Model	Manifest + Runtime	Runtime + Entitlements
Code Signing	APK signing	Mandatory code signing
SELinux/Sandboxing	SELinux policies	TrustedBSD MAC framework

8. PERFORMANCE OPTIMIZATION

Process Management Best Practices:

For Android:

- Use bound services for IPC instead of AIDL when possible
- Implement proper service lifecycle management
- Use JobScheduler for background tasks
- Minimize process creation overhead

For iOS:

- Use App Extensions instead of background processing
- Implement proper XPC service lifecycle
- Use GCD and Operation Queues effectively
- Minimize memory footprint in background

Performance Monitoring:

Metric	Android Tools	iOS Tools
Memory Usage	Memory Profiler	Instruments (Allocations)
CPU Usage	Systrace	Instruments (Time Profiler)
IPC Performance	Method Tracing	Instruments (System Trace)
Battery Impact	Battery Historian	Instruments (Energy Log)