MODULE 1

Introduction to Operating System

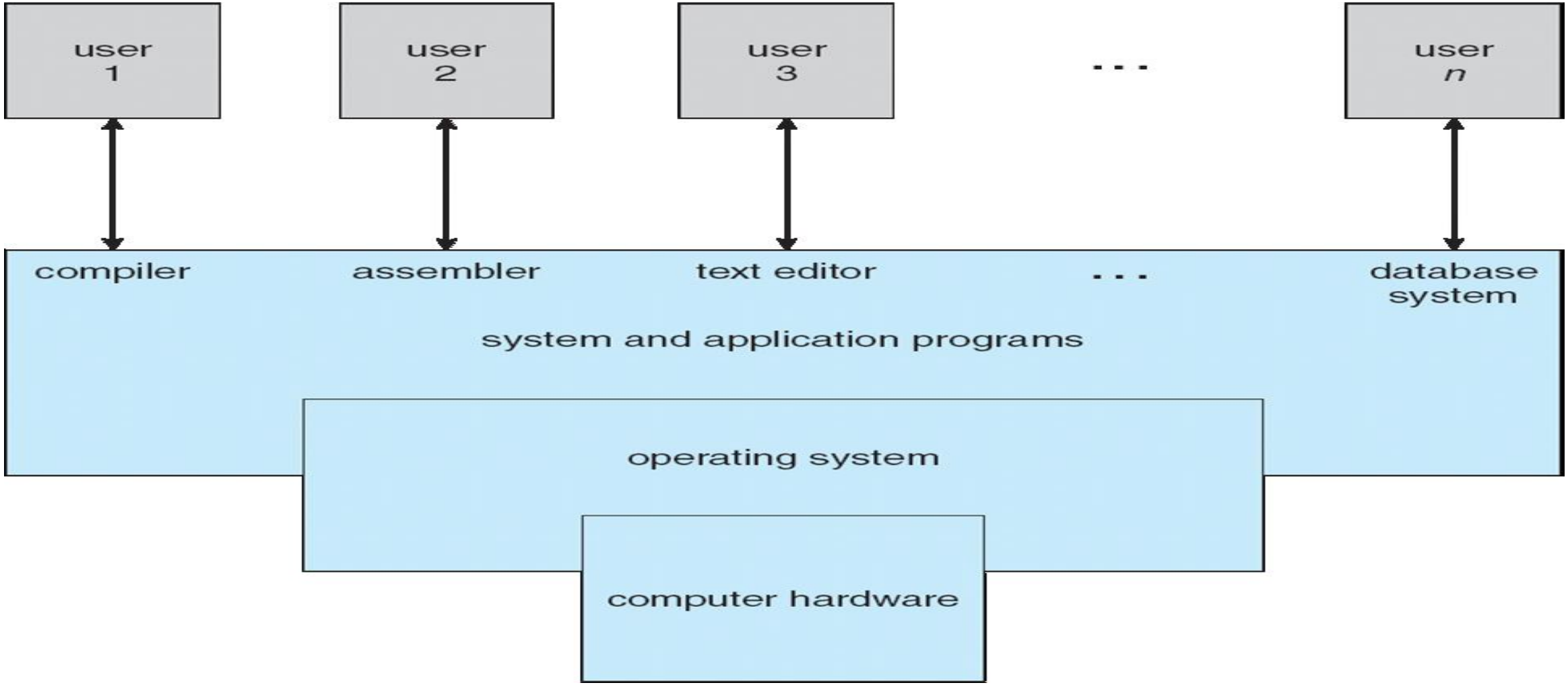
- Introduction to Operating System: Basics of Operating Systems: Definition, Operations Dual-Mode, and Multi-Mode, Services, System Call Types.
- Operating System Structure: Layered Structure, Microkernel's, modules.
- Hybrid Systems- Mac OS X, iOS, Android

Dr. Renuka S. Gound

Introduction: Computer System Structure

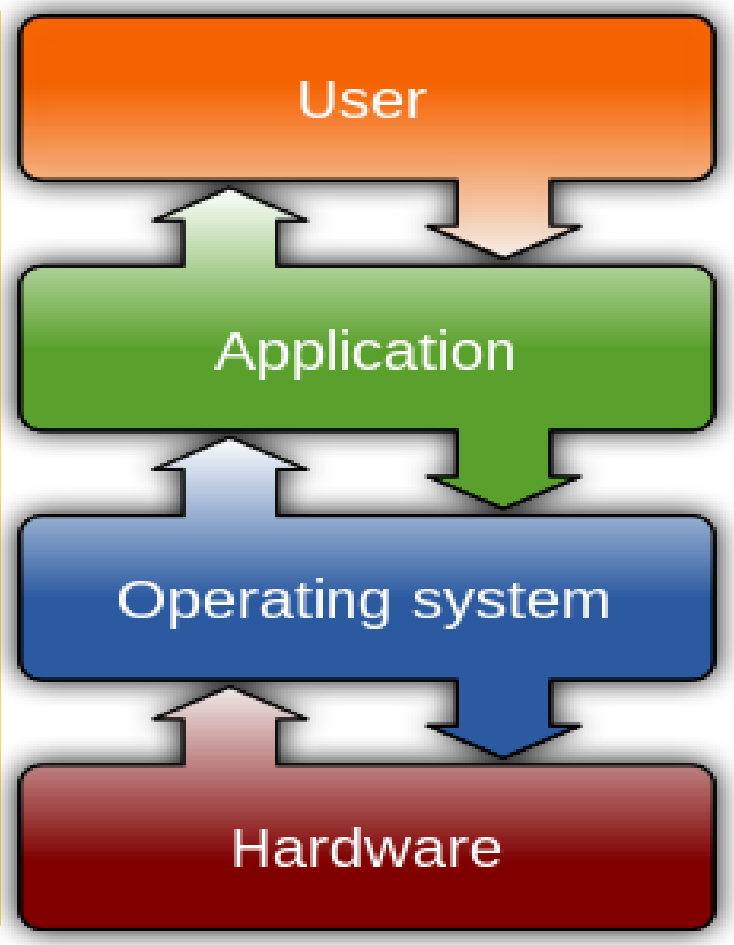
- Computer system can be divided into four components:
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates the use of the hardware among various applications and users
  - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, and other computers

Introduction: Four Components of a Computer System



Introduction: What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner



Introduction: Operating System Definition

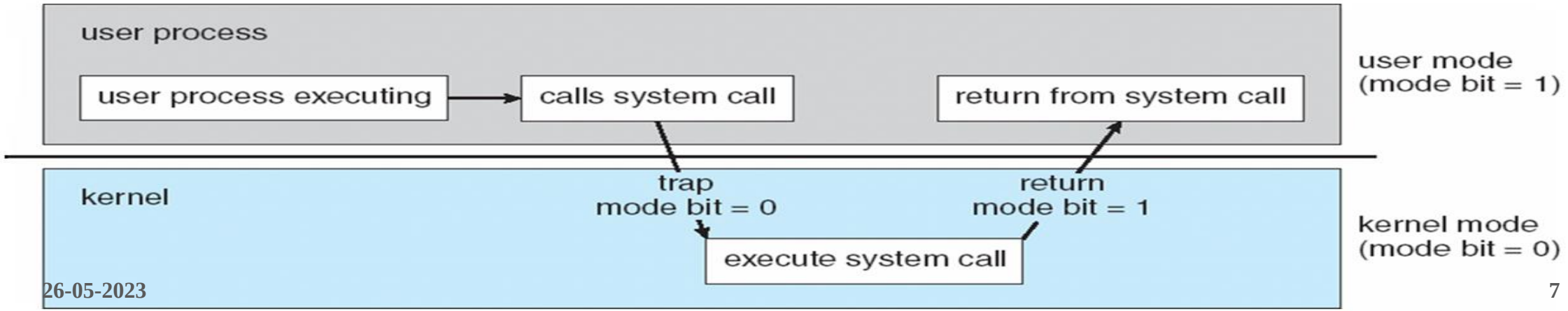
- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

Operations Dual-Mode and Multi-Mode

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - Provides the ability to distinguish when the system is running user code or kernel code
    - Some instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to the kernel, return from call resets it to user
- Increasingly CPUs support multi-mode operations
  - i.e. **virtual machine manager** (**VMM**) mode for guest **VMs**

Transition from User to Kernel Mode

- Timer to prevent infinite loop/process hogging resources
  - Timer is set to interrupt the computer after some time period
  - Keep a counter that is decremented by the physical clock.
  - Operating system set the counter (privileged instruction)
  - When counter zero generates an interrupt
  - Set up before scheduling process to regain control or terminate a program that exceeds the allotted time



Operating System Services

- Operating systems provide an environment for the execution of programs and services to programs and users
- One set of operating-system services provides functions that are helpful to the user:
- **User interface -** Almost all operating systems have a user interface (UI). Varies between Command-Line (CLI), Graphics User Interface (GUI), Batch
- **Program execution -** The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
- **I/O operations -** A running program may require I/O, which may involve a file or an I/O device

Operating System Services (Cont.)

- **File-system manipulation -** The file system is of particular interest. Programs need to read and write files and directories, create and delete them, search them, list file Information, and permission management.
- **Communications –** Processes may exchange information, on the same computer or between computers over a network Communications may be via shared memory or through message passing (packets moved by the OS)
- **Error detection –** OS needs to be constantly aware of possible errors that May occur in the CPU and memory hardware, in I/O devices, in the user program

Operating System Services (Cont.)

- For each type of error, OS should take the appropriate action to ensure correct and consistent computing Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system
- Another set of OS functions exists for ensuring the efficient operation of the **system** itself via resource sharing
- **Resource allocation -** When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
- Many types of resources - CPU cycles, main memory, file storage, and I/O devices.

Operating System Services (Cont.)

- **Accounting -** To keep track of which users use how much and what kinds of computer resources
- **Protection and security -** The owners of information stored in a multiuser or networked computer system may want to control the use of that information, concurrent processes should not interfere with each other
- Protection involves ensuring that all access to system resources is controlled Security of the system from outsiders requires user authentication and extends to defending external I/O devices from invalid access attempts

Types of System Calls in OS

1. Process Control

- To forcefully abort the process, simply end it normally.
- Execute a process after loading it into the main memory.
- Terminate the current process before starting a new one.
- Wait for a process to complete running. Wait until a specific event happens, then announce it once it has.
- Allocate memory to a process, then release the memory if the process is terminated.

2. File Management

- Making and erasing files
- Open the file, then close it.
- Write to a specific file, and read from a specific file.
- To obtain a file's attribute and to change a file's attribute

3.Device Management

- Devices might be needed while a process is running. such as access to the file system, I/O devices, main memory, etc.
- As a result, it can ask for a device and then release it once the task is complete.
- when a requested device is granted access by the process. It is capable of reading, writing, and repositioning operations.
- In order to obtain or modify a specific device's attribute.
- To detach a device from the processor that is currently executing a command, the call can be made.

4. Information Maintenance

- Obtain the system's time or date. Set the system's time or date.
- Obtain system-related information. Configure the system data.
- Obtain the characteristics of a specific operating system process. Alternatively, of a specific file on the system or on any attached devices.
- Set the characteristics of a specific operating system process. Alternatively, of a specific file on the system or on any attached devices.

5.Communication

- Open a fresh connection to send the data. After the transmission is finished, disconnect from the connection.
- On a particular connection, send a message. Obtain communication from a specific connection.
- Identify and connect a specific remote device to the network. Remove a specific remote computer or device from the network.

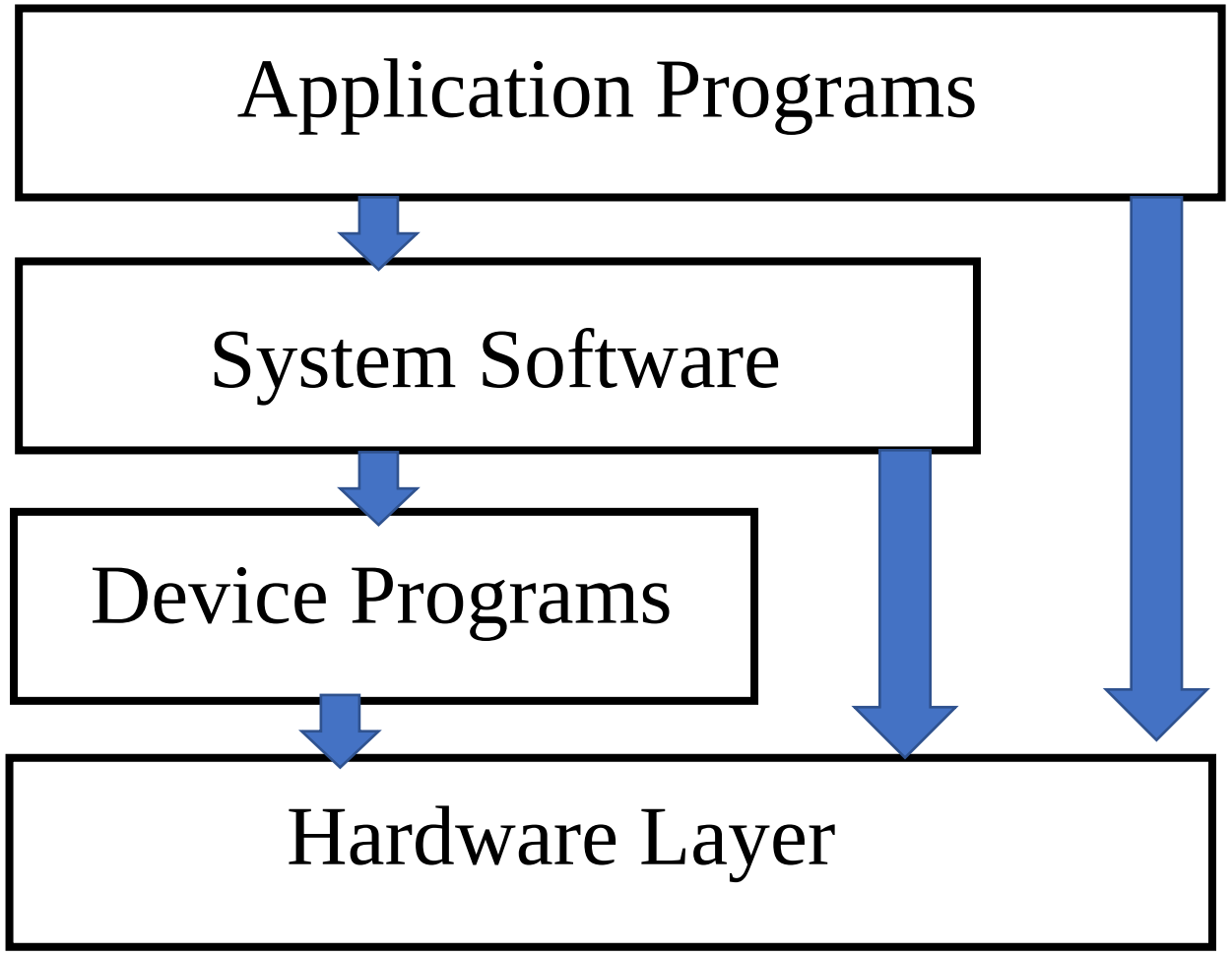
Example of System Calls in Windows and Unix

Types of System Calls	Windows	Linux
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Management	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()

Example of System Calls in Windows and Unix

Types of System Calls	Windows	Linux
Device Management	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()

Structure of Operating System

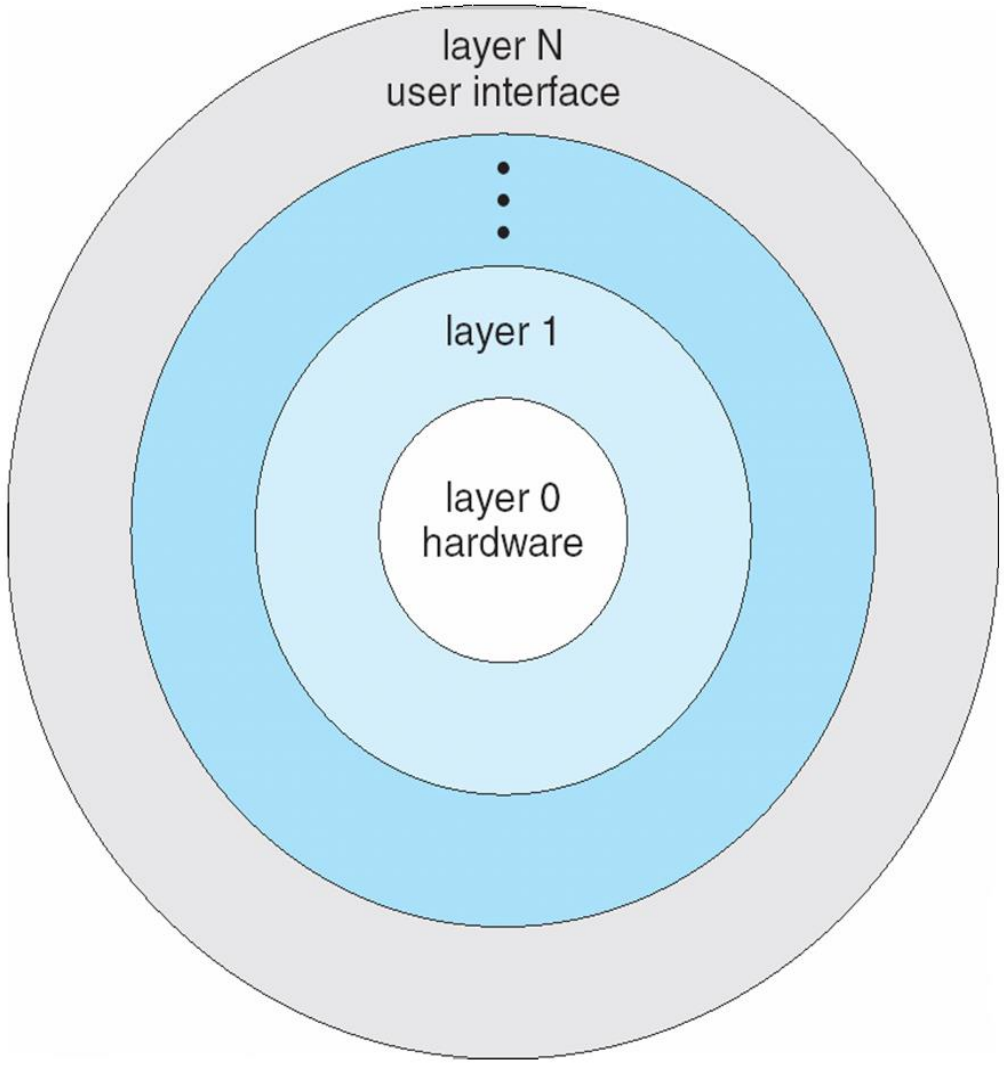


- Each layer is having access to the hardware layer.
- The interfaces and level of functionalities are not well separated.
- This type of structure makes the operating system vulnerable to errant and malicious programs.
- **Example: MS-DOS**

Fig. Simple Structure of OS

Architecture of Layered Structure

- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers
- Abstraction: Every layer is concerned with its functions. So the functions and implementations of the other layers are abstract to it.



Advantages of Layered Structure

- **Modularity:** This design promotes modularity as each layer performs only the tasks it is scheduled to perform.
- **Easy debugging:** As the layers are discrete so it is very easy to debug. Suppose an error occurs in the CPU scheduling layer. The developer can only search that particular layer to debug, unlike the Monolithic system where all the services are present.
- **Easy update:** A modification made in a particular layer will not affect the other layers.
- **No direct access to hardware:** The hardware layer is the innermost layer present in the design. So a user can use the services of hardware but cannot directly modify or access it, unlike the Simple system in which the user had direct access to the hardware.

Disadvantages of Layered Structure

- **Complex and careful implementation:** As a layer can access the services of the layers below it, so the arrangement of the layers must be done carefully. For example, the backing storage layer uses the services of the memory management layer. So it must be kept below the memory management layer. Thus with great modularity comes complex implementation.
- **Slower in execution:** If a layer wants to interact with another layer, it requests to travel through all the layers present between the two interacting layers. Thus it increases response time. Thus an increase in the number of layers may lead to a very inefficient design.
- **Functionality:** It is not always possible to divide the functionalities. Many times, they are interrelated and can't be separated.

Microkernel System Structure

- Plugin Architecture: Computer software that adds new functions to a host program without altering the host program itself
- **Description**
- The plug-in architecture consists of two components: a core system and plug-in modules.
- The main key design here is to allow adding additional features as plugins to the **core application,** providing extensibility, flexibility, and isolation of application features and customs processing logic.
- The specific rules and processing are separate from the core system. At any given point, we can add, remove, and change existing **plugins** with little or no effect on the rest of the core system or other plug-in modules.

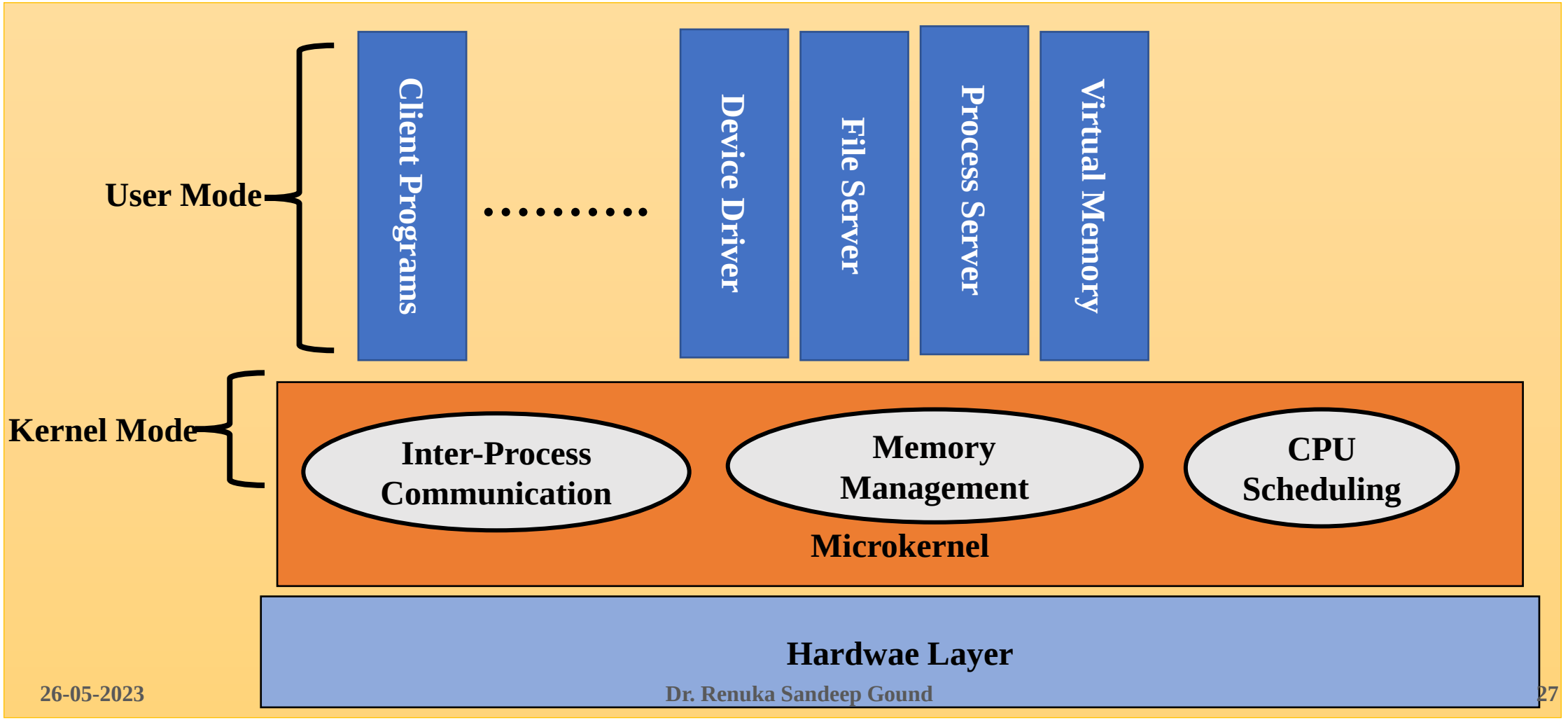
Microkernel Approach

- In the mid-1980s, researchers at Carnegie Mellon University developed an operating system called Mach that modularized the kernel using the microkernel approach.
- This method structures the operating system by removing all nonessential components from the kernel and implementing them as user-level programs that reside in separate address spaces. The result is a smaller kernel
- The main function of the microkernel is to provide communication between the client program and the various services that are also running in user space. Communication is provided through message passing.

Benefits of using Micro-Kernel Structure

- It makes **extending the operating system easier**. All new services are added to user space and consequently do not require modification of the kernel. When the kernel does have to be modified, the changes tend to be fewer, because the microkernel is a smaller kernel.
- **The resulting operating system is easier to port from one hardware design to another.**
- The microkernel also **provides more security and reliability**, since most services are running as user rather than kernel processes. If a service fails, the rest of the operating system remains untouched.

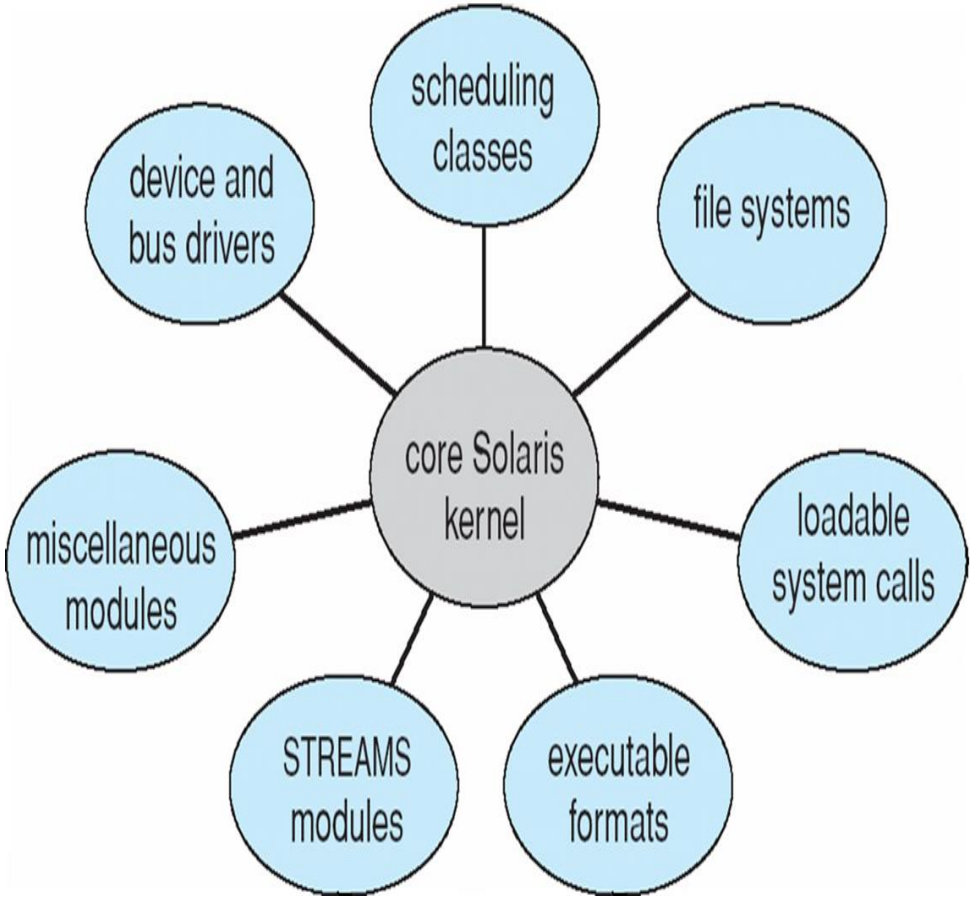
Micro-Kernel Architecture



Modules

- Many modern operating systems implement loadable kernel modules
- Uses an object-oriented approach
- Each core component is separate
- Each talk to the others over known interfaces
- Each is loadable as needed within the kernel
- Overall, similar to layers but with more flexible
- Linux, Solaris, etc

Solaris Modular Approach



Modules

- operating-system design involves using loadable kernel modules (LKMs)..
- The idea of the design is for the kernel to provide core services, while other services are implemented dynamically, as the kernel is running.
- Linking services dynamically is preferable to adding new features directly to the kernel, which would require recompiling the kernel every time a change was made.
- Thus, for example, it needs to build CPU scheduling and memory management algorithms directly into the kernel and then add support for different file systems by way of loadable modules.

Modules

- The overall result resembles a layered system in that each kernel section has defined, protected interfaces; but it is more flexible than a layered system because any module can call any other module.
- The approach is also similar to the microkernel approach in that the primary module has only core functions and knowledge of how to load and communicate with other modules; but it is more efficient because modules do not need to invoke message passing in order to communicate.

Hybrid Systems

- In practice, very few operating systems adopt a single, strictly defined structure. Instead, they combine different structures, resulting **in hybrid systems** that address performance, security, and usability issues.
- **Linux and Solaris kernels** in kernel address space, so **monolithic, plus modular** for dynamic loading of functionality
- **Windows** is mostly **monolithic, plus microkernel** for different subsystem personalities

Apple’s macOS and ios

- Apple’s macOS operating system is designed to run primarily on desktop and laptop computer systems, whereas iOS is a mobile operating system designed for the iPhone smartphone and iPad tablet computer. Architecturally, macOS and iOS have much in common
- Therefore they can be presented together, highlighting what they share as well as how they differ from each other.

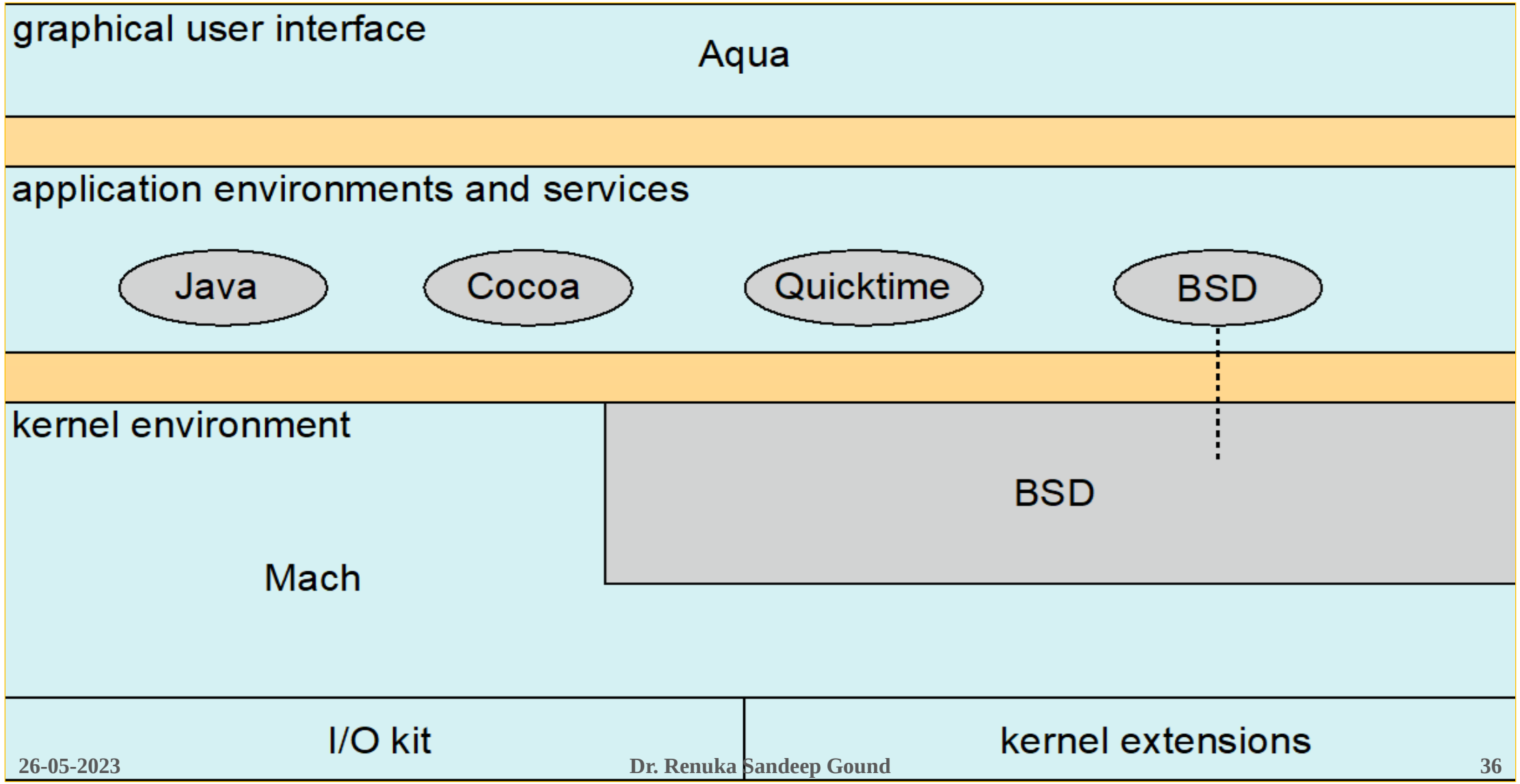
Some significant distinctions between macOS and iOS

- Because macOS is intended for desktop and laptop computer systems, it is compiled to run on Intel architectures. iOS is designed for mobile devices and thus is compiled for ARM-based architectures.
- Similarly, the iOS kernel has been modified somewhat to address specific features and needs of mobile systems, such as power management and aggressive memory management.
- Additionally, iOS has more stringent security settings than macOS. The iOS operating system is generally much more restricted to developers than macOS and may even be closed to developers.
- For example, iOS restricts access to POSIX and BSD APIs on iOS, whereas they are openly available to developers on macOS

Components of the Mac OS X Structure

- Details about the different components of the Mac OS X structure as seen in the image are as follows −
- **Core OS**
- The Darwin Core is based on the BSD (Berkeley Software Distribution) version of Unix.
- Mach is the main part of the Darwin core and it performs operations such as memory use, data flow from and to CPU etc.
-

Mac OS X Structure



Mac OS X Structure

- Darwin is also open source i.e. anyone can obtain its source code and make modifications to it. Different versions of Darwin can be used to enhance the Mac OS X.
- Some of the major features of the Darwin core are protected memory, automatic memory management, preemptive multitasking, advanced virtual memory etc
- It also provides I/O services for Mac OS X and supports plug-and-play, hot-swapping and power management.

Graphics Subsystem

- The graphics subsystem in the Mac OS X contains three parts i.e. Quartz, OpenGL, and QuickTime.
- The 2-D graphics in the graphics subsystem is managed by **Quartz**. It provides fonts, interface graphics, rendering of the images etc.
- **OpenGL** provides support for 3-D graphics in the system such as texture mapping, transparency, antialiasing, atmospheric effects, special effects etc. It is also used in Unix and Windows systems.
- **QuickTime** is used for different digital media such as digital video, audio and video streaming etc. It also enables creative applications such as iMovie, iTunes etc.

Application Subsystem

- The application subsystem in Mac OS X provides the classic environment to run classic applications.
- **Carbon, Cocoa and Java** are the three application development environments available.
- The classic environment makes sure that applications written for the previous versions of the operating system can run smoothly.
- The carbon environment is used to port existing applications to carbon application program interfaces. This is called carbonising the application.
- The cocoa environment provides object-oriented application development environment. The cocoa applications use the benefits of the Mac OS X Structure the most.
- The Java applications and Java applets can be run using the Java environment.

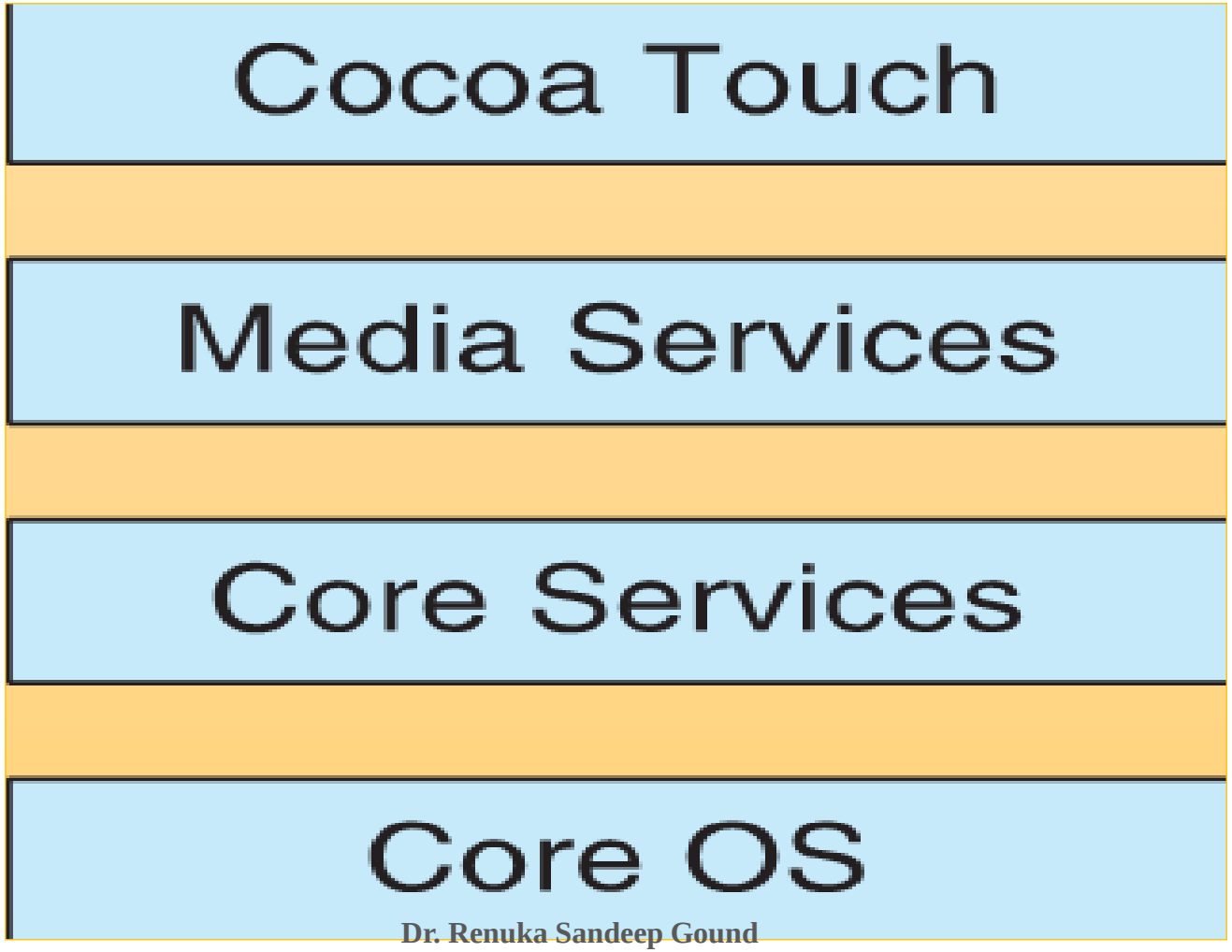
User Interface

- Aqua is the user interface of Mac OS X.
- It provides good visual features as well as the tools to customize the user interface as per the user requirements.
- Aqua contains extensive use of color and texture as well as extremely detailed icons. It is both pleasant to view and efficient to use.

iOS

- Apple mobile OS for iPhone, iPad
- Structured on Mac OS X, added functionality
- Does not run OS X applications natively
- Also runs on different CPU architectures (ARM vs. Intel)
- Cocoa Touch Objective-C API for developing apps
- Media services layer for graphics, audio, video
- Core services provide cloud computing, databases
- Core operating system, based on Mac OS X kernel

iOS Architecture



CORE OS Layer:

All the IOS technologies are built under the lowest level layer i.e. Core OS layer. These technologies include:

1. Core Bluetooth Framework
2. External Accessories Framework
3. Accelerate Framework
4. Security Services Framework
5. Local Authorization Framework etc.
6. It supports 64 bit which enables the application to run faster.

CORE SERVICES Layer:

- Some important frameworks are present in the CORE SERVICES Layer which helps the iOS operating system to cure itself and provide better functionality. It is the 2nd lowest layer in the Architecture as shown above. Below are some important frameworks present in this layer:
- **Address Book Framework-**
- The Address Book Framework provides access to the contact details of the user.
- **Cloud Kit Framework-**
- This framework provides a medium for moving data between your app and iCloud.
- **Core Data Framework-**
- This is the technology that is used for managing the data model of a Model View Controller app.

Core Foundation Framework-

This framework provides data management and service features for iOS applications.

Core Location Framework-

This framework helps to provide the location and heading information to the application.

Core Motion Framework-

All the motion-based data on the device is accessed with the help of the Core Motion Framework.

Foundation Framework-

Objective C covering too many of the features found in the Core Foundation framework.

HealthKit Framework-

This framework handles the health-related information of the user.

HomeKit Framework-

This framework is used for talking with and controlling connected devices with the user's home.

Social Framework-

It is simply an interface that will access users' social media accounts.

StoreKit Framework-

This framework supports for buying of contents and services from inside iOS apps.

MEDIA Layer:

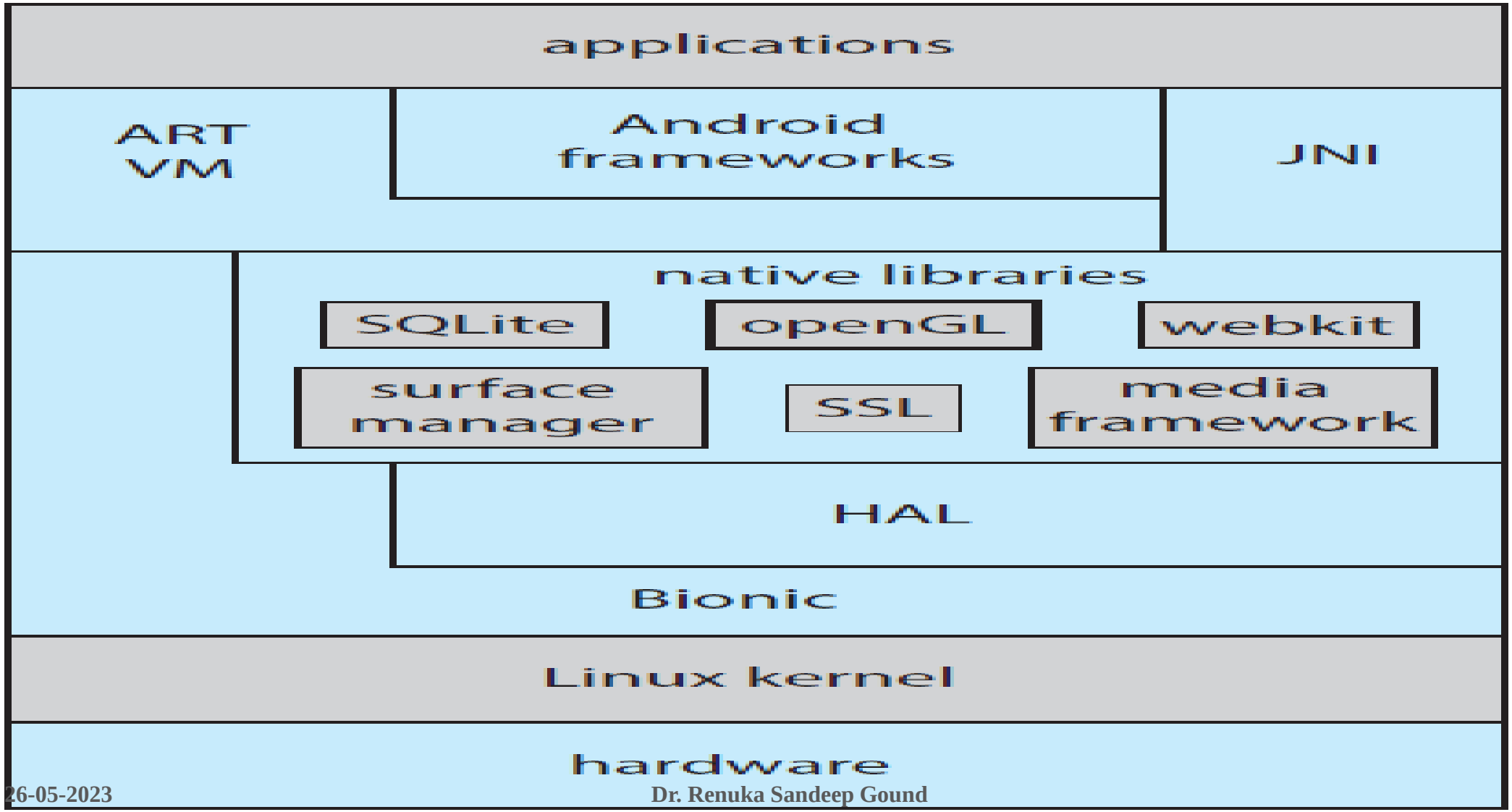
- With the help of the media layer, we will enable all graphics video, and audio technology of the system. This is the second layer in the architecture. The different frameworks of MEDIA layers are:
- **ULKit Graphics-**
- This framework provides support for designing images and animating the view content.
- **Core Graphics Framework-**
- This framework support 2D vector and image-based rendering and it is a native drawing engine for iOS.
- **Core Animation-**
- This framework helps in optimizing the animation experience of the apps in iOS.

- **Media Player Framework-**
- This framework provides support for playing the playlist and enables the user to use their iTunes library.
- **AV Kit-**
- This framework provides various easy-to-use interfaces for video presentation, recording, and playback of audio and video.
- **Open AL-**
- This framework is an Industry Standard Technology for providing Audio.
- **Core Images-**
- This framework provides advanced support for motionless images.
- **GL Kit-**
- This framework manages advanced 2D and 3D rendering by hardware-accelerated interfaces.

COCOA Touch

- It is also known as the application layer which acts as an interface for the user to work with the iOS Operating system. It supports touch and motion events and many more features. The COCOA TOUCH layer provides the following frameworks :
- **EvenKit Framework-**This framework shows a standard system interface using view controllers for viewing and changing events.
- **GameKit Framework-**This framework provides support for users to share their game-related data online using a Game Center.
- **MapKit Framework-**This framework gives a scrollable map that one can include in your user interface of the app.
- **PushKit Framework-**This framework provides registration support.

Android Architecture



Android

- The Android operating system was designed by the Open Handset Alliance (led primarily by Google) and was developed for Android smartphones and tablet computers.
- Android is a layered stack of software that provides a rich set of frameworks supporting graphics, audio, and hardware features.
- Software designers for Android devices develop applications in the Java language, but they do not generally use the standard Java API.

Android

- Google has designed a separate Android API for Java development. Java applications are compiled into a form that can execute on the Android Run Time (ART),
- ART is a virtual machine designed for Android and optimized for mobile devices with limited memory and CPU processing capabilities.
- Java programs are first compiled to a Java bytecode .class file and then translated into an executable .dex file.
- Whereas many Java virtual machines perform just-in-time (JIT) compilation to improve application efficiency,

JNI

- Android developers can also write Java programs that use the Java native interface (JNI) which allows developers to bypass the virtual machine and instead write Java programs that can access specific hardware features.
- Programs written using JNI are generally not portable from one hardware device to another
- The set of native libraries available for Android applications includes frameworks for developing web browsers (webkit), database support (SQLite), and network support, such as secure sockets (SSLs).

Hardware abstraction layer (HAL)

- Because Android can run on an almost unlimited number of hardware devices,
- Google has chosen to abstract the physical hardware through the hardware abstraction layer, or HAL.
- By abstracting all hardware, such as the camera, GPS chip, and other sensors, the HAL provides applications with a consistent view independent of specific hardware.
- This feature, of course, allows developers to write programs that are portable across different hardware platforms.

Bionic

- The standard C library used by Linux systems is the GNU C library (glibc).
- Google instead developed the Bionic standard C library for Android.
- Not only does Bionic have a smaller memory footprint than glibc, but it also has been designed for the slower CPUs that characterize mobile devices.
- (In addition, Bionic allows Google to bypass GPL licensing of glibc.)

Linux kernel

- At the bottom of Android's software stack is the Linux kernel.
- Google has modified the Linux kernel used in Android in a variety of areas to support the special needs of mobile systems, such as power management.
- It has also made changes in memory management and allocation and has added a new form of IPC known as Binder

Android

- Developed by Open Handset Alliance (mostly Google)
- Open Source
- Similar stack to IOS
- Based on Linux kernel but modified
- Provides process, memory, device-driver management
- Adds power management
- Runtime environment includes core set of libraries and Dalvik virtual machine Apps developed in Java plus Android API
- Java class files compiled to Java bytecode then translated to executable than runs in Dalvik VM
- Libraries include frameworks for web browser (webkit), database (SQLite), multimedia, smaller libc