Indira Gandhi
National Open University
School of Computer and
Information Sciences

MCS-203
Operating Systems

Block

4

CASE STUDIES

PROGRAMME/COURSE DESIGN COMMITTEE

Prof (Retd) S. K. Gupta,
IIT Delhi

Prof. T.V. Vijaya Kumar, Dean,
SC&SS, JNU,
New Delhi

Prof. Ela Kumar, Dean, CSE, IGDTUW,
Delhi

Prof. Gayatri Dhingra,
GVMITM,
Sonipat, Haryana

Sh. Milind Mahajani Vice President,
Impressico Business Solutions
Noida, UP

Prof. V. V. Subrahmanyam, Director
SOCIS, New Delhi

Prof. P. V. Suresh,
SOCIS, IGNOU, New Delhi

Dr. Shashi Bhushan
SOCIS, IGNOU, New Delhi

Shri Akshay Kumar, Associate Prof.
SOCIS, IGNOU, New Delhi

Shri M. P. Mishra, Associate Prof.
SOCIS, IGNOU, New Delhi

Dr. Sudhansh Sharma, Asst. Prof
SOCIS, IGNOU, New Delhi

BLOCK PREPARATION TEAM

Prof. D.P. Vidyarthi *(Content Editor)*
Jawaharlal Nehru University
New Delhi

Prof. K. Swathi *(Course Writer- Units 2 & 3)*
NRIIT, Vijayawada (Rural), AP

Miss Jyoti Bisht *(Course Writer – Units 1 & 4)*
Research Scholar, SOCIS, IGNOU,
New Delhi

Prof. V.V. Subrahmanyam
SOCIS, IGNOU

Prof. Parmod Kumar *(Language Editor)*
School of Humanities
IGNOU

Course Coordinator: Prof. V. V. Subrahmanyam

PRINT PRODUCTION

Mr. Tilak Raj
Assistant Registrar,
MPDD, IGNOU, New Delhi

July, 2021

ISBN : 978-93-91229-15-3

Further information on the Indira Gandhi National Open University courses may be obtained from the University's office at Maidan Garhi, New Delhi-110068.

Printed and published on behalf of the Indira Gandhi National Open University, New Delhi by the Registrar, MPDD, IGNOU, New Delhi

Laser Typesetting : Akashdeep Printers, 20-Ansari Road, Daryaganj, New Delhi-110002

Printed at : Akashdeep Printers, 20-Ansari Road, Daryaganj, New Delhi-110002

BLOCK INTRODUCTION

This block introduces you to the Case Studies on Windows 10, Linux, Android and iOS.

The block is organized into 4 units.

Unit 1 covers the Windows 10

Unit 2 covers the LINUX

Unit 3 covers one of the most popular mobile operating system; the Android

Unit 4 covers another most popular mobile operating system; the iOS.

ignou
THE PEOPLE'S
UNIVERSITY

UNIT 1 CASE STUDY - WINDOWS 10

Structure

1.0 INTRODUCTION

An operating system is set of programs that act as an interface between user and hardware. It is a type of system software that provides various low level functionalities like process management, memory management, I/O management, file management, scheduling. Examples of popular operating system are Windows, Linux, Unix, Android, iOS, Mac.

Operating system is used to perform two major activities-

- Interfacing user with hardware
- Management of Resources

Windows 10 is one of the most popular and latest operating system of Microsoft. It is introduced with the concept of "Universal apps", which provides apps designed to work for multiple platforms like personal computers, smartphones, Xbox One consoles, laptops and other compatible devices. It is designed for various platforms.

Windows is widely used operating system for desktop users. The various reasons for its popularity among users are-

- Easy to use interface
- High availability of compatible software
- Provides support for new hardware
- Provides Plug and Play feature

Although it is popular among desktop users, it has several disadvantages over other operating systems.

- It is proprietary
- It is closed source
- It is susceptible to virus hence not secure
- Lacks good technical support
- Resource requirements are high

In this unit, we will study about the features, evolution, process management, memory management and file management in Windows 10 Operating System.

1.1 OBJECTIVES

After going through this unit, you should be able to:

- Know about the features of Windows 10
- Understand various editions of Windows 10 operating system
- List various components of Windows
- Describe about Process and memory management in Windows 10
- Describe how files and disk are managed
- Know about the compression used in Windows 10

1.2 FEATURES OF WINDOWS 10

Windows 10 have all the traditional features of other windows version with a wide variety of new features added to it. Features of Windows 10 are:

1.2.1 Multi user – it can be accessed by multiple users simultaneously.

1.2.2 Multi tasking – can perform multiple tasks at the same time.

1.2.3 Resource sharing – files, folders, drivers etc can be shared.

1.2.4 Different versions are available for different types of devices.

1.2.5 Universal apps – designed to run on multiple compatible Microsoft products.

1.2.6 Cortana – a voice controlled digital assistant included in desktop

- Xbox app – can capture and share Xbox games
- Microsoft web browser
- Continuum tablet mode – for touch sensitive devices
- Task view – a desktop management feature
- Windows Hello Face, Hello fingerprint, Hello PIN, security key, picture password sign-in options available.
- Dynamic lock – locks your PC when paired device goes out of range.
- DirectX 12 – used for handling multimedia tasks.
- New builds are available to users at no additional cost.

1.3 EVOLUTION OF WINDOWS 10

It's been over 35 years, since Windows 1.0 was released, and now Microsoft released Windows 10.

It all started in November 1985 when the first version of Windows, **Windows 1.0** was made available. It required two floppy disks and 192KB of RAM to install and was essentially a front-end for MS-DOS, creating a graphical environment for the platform and capable of summoning certain functions.

Windows 2.0(in the year 1987) built on its predecessor by adding more sophisticated keyboard shortcuts, overlapping windows and support for VGA graphics and introducing features that have become staples of the Windows experience. These include the control panel, the terms *minimise* and *maximise* and support for the first Windows versions of Word and Excel. Windows 2.0 was also the last version not to need a hard disk to install.Six months later, **Windows 2.1** was released to take advantage of new Intel processors, while a further update in 1989 made minor changes to memory management, printing and drivers.

The third major release of **Windows 3.0** (in the year 1990) was arguably the first successful one, with a revamped UI, better memory management, 256 colour VGA mode and enhanced multimedia options such as support for sound cards and CD-ROMS. Notably, Windows 3.0 also separated files and applications into a list-based manager for the former and an icon-based manager for the latter.

At this point, although Microsoft didn't plan for it, the Windows line divided into the NT and 9x families. The business-focused NT 3.1, the first in the line, was built from scratch as a 32-bit operating system made to have the same aesthetics as the 16-bit Windows 3.1. It was released in July 1993 in workstation and server editions, and received an incremental update in the form of **Windows NT 3.5**. **Windows NT 4.0** (right) followed three years later, boasting a similar interface to Windows 95, although NT 4.0 was considered much more stable than its consumer counterpart.

Windows 95(in the year 1995) was originally meant to be based on NT architecture but Microsoft abandoned these plans amid fears an NT-based consumer operating system wouldn't run on low-end hardware and would take too long to develop, opening up a gap in the market that could be seized by others. This need to reach the shelves quickly resulted in the Windows 9x family, the first entry of which was Windows 95.

Backed by a huge marketing campaign, Windows 95 came with a significantly enhanced user interface which introduced the start button and taskbar. It also added simpler plug and play features and although the kernel itself was 32-bit, much of the code remained 16-bit because it was recycled from 3.1.

Windows 98 (in the year 1998) a follow-up was released three years later and was considered to be more stable than Windows 95, with easier support for plug and play devices, larger disk partitions, more robust USB functionality, networking improvements and new system tools.

Windows ME was intended as a stopgap release between 98 and XP and it showed. It became notorious for instability and unreliability, but at least it introduced the 'system restore' feature to give users a fighting chance of getting any work done. Many of the new features had already been made available to Windows 95 and 98 users through updates and many enterprise functions were absent as Microsoft attempted to make a clear distinction between the consumer-facing ME and business-focused Windows 2000.

Later on **WINDOWS XP** was released in the year 2001, **Windows Vista** in 2007, **Windows 7** in 2009 and **Windows 8** in 2012. **Windows 10** is an operating system of Windows NT family developed by Microsoft Corporation. It was released in 2015 as a successor to Windows 8.

1.4 WINDOWS 10 EDITIONS

For various devices, windows 10 have total 12 editions with different set of features. Some are available only through volume licensing. Some of the major editions are-

Windows 10 Home

It is designed for PC, tablets, 2 in 1 PCs. It includes features like- Universal Windows app, Cortana assistance, Continuum tablet mode, Xbox, Microsoft edge browser, biometric sign-in.

Windows 10 Pro

It has all the features present in Windows 10 Home along with additional features like –Remote desktop, Bit Locker encryption, Hyper-V, group Policy, support for active directory, windows update for business.

Windows 10 Enterprise

It provides all the features of Windows 10 Pro along with advanced security and management needs like- Resilient file system, windows defender credential guard, application guard and advanced threat protection.

Windows 10 Education

It provides features similar to Windows Enterprise but provides tools explicitly for academic purpose.

Windows 10 Mobile

It is designed for smart phones, tablets, touch screen devices but later Microsoft discontinued this due to customer dissatisfactions.

This edition is specially designed for IoT (internet of Things) devices. It provides security, manageability to Iot devices along with windows ecosystem and connectivity with cloud.

1.5 WINDOWS OS COMPONENTS

Windows operating system has two modes for its operation – kernel mode and User mode. User applications run in user mode and system operations runs in kernel mode. Applications that run in kernel mode shares single virtual address space while a different private virtual address space is allocated to each application in user mode.

When an application is started in user mode, a process is created by windows and a private virtual address space is allocated. Since the address space is private, applications cannot access or modify data belonging to other applications. Similarly they also cannot access virtual address reserved for operating system. Also if due to some reason an application crashes, operating system and other application will not be affected.

On the other hand, single virtual address space is shared by processes running in kernel mode. It means that processes are not isolated by each other. Drivers are usually run in kernel mode. Internal components of windows operating system is shown in figure 1.

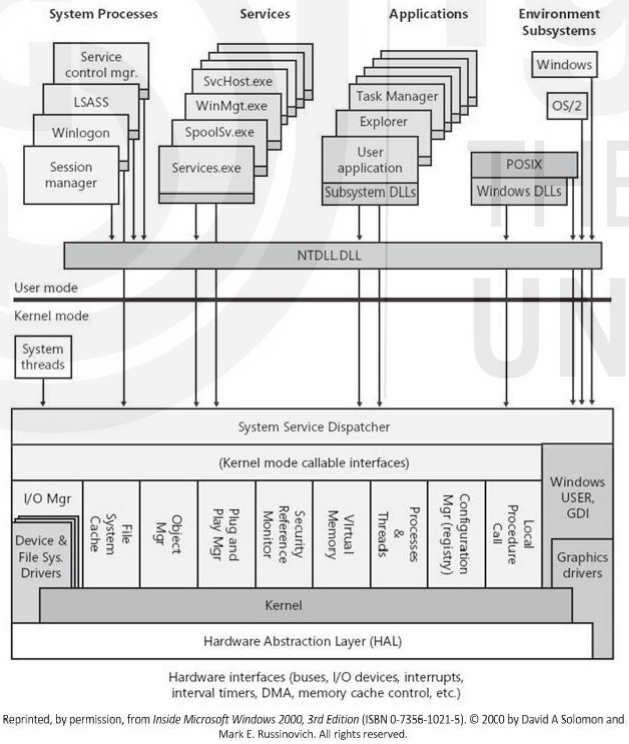

Fig 1: Windows Components (*Source: docs.microsoft.com*)

Universal Windows Platform Apps

Universal Windows Platform (UWP) is introduced in Windows 10. It is common platform that can be used for different types of devices that runs windows 10. In this same core API is used for all devices. The apps are written in C++/WinRT or C++/CX and can access Win32 API which is implemented by all window 10 devices. For user interface DirectX, XAML or HTML is used. If apps need to be designed for taking advantage of specific device than extension SDK can be used to call specific APIs.

Fig 2: Windows 10 Universal Apps Platform (*Source: docs.microsoft.com*)

1.6 PROCESS MANAGEMENT IN WINDOWS 10

1.6.1 Processes and Threads

A program in execution is called a process. A process contains information like- unique process identifiers, virtual address space, priority, environment variables, working set size, executable code and security context. Initially process starts execution with a single thread, known as primary thread but later additional threads can be created by *primary thread*. Every thread maintains information like unique thread identifier, priority, local storage, data structure to save thread context and security context. Thread context contains user stack, kernel stack, and set of registers. All threads of a single process shares system resources and virtual address space.

In windows multiple threads belonging to different processes can be executed simultaneously and creates the effect of simultaneous execution for single processor system. Hence it is called multitasking. Also windows follows preemptive approach of scheduling hence time slice is allocated to each thread. After the time slice elapses, switching is done to another thread and system saves the context of preempted thread. Context of new thread is loaded for execution.

1.6.2 Scheduling

Scheduling decides the order in which threads will be executed. Scheduling of threads is done by giving priorities in range the between 0 to 31, where 0 represents minimum and 31 represents maximum priority. Threads with same priority are treated as equal. Time slices is assigned to thread in round-robin fashion to high priority threads. Factors to determine priority of thread are:

1. Priority class of process – process can belong to any of the class - IDLE_PRIORITY_CLASS, BELOW_NORMAL_PRIORITY_CLASS, NORMAL_PRIORITY_CLASS, ABOVE_NORMAL_PRIORITY_CLASS, HIGH_PRIORITY_CLASS, REALTIME_PRIORITY_CLASS

2. Priority level of thread – priority levels of thread within a process can be any one of the following- THREAD_PRIORITY_IDLE,
   THREAD_PRIORITY_LOWEST
   THREAD_PRIORITY_BELOW_NORMAL,
   THREAD_PRIORITY_NORMAL
   THREAD_PRIORITY_ABOVE_NORMAL,
   THREAD_PRIORITY_HIGHEST
   THREAD_PRIORITY_TIME_CRITICAL

Priority of a thread is formed by combining these to priorities- priority class of process and priority level of thread. This is called **base priority** of thread.

Priority Boosts

When threads of higher priority gets execute every time, the system can suffer form the problem of starvation. To solve this problem Windows uses the concept of dynamic priorities. Initially priority of each thread is initialized with base priority. The system can later change the priority of threads so that no thread gets starved. The threads having base priority between 0 to 15 can receives priority boosts. Each time when a thread completes a time slice, its priority is reduced by one level by the scheduler until thread gets its base priority back. Once the base priority is reached, the priority is not changed further.

User mode scheduling

Applications can also schedule their own threads using user mode scheduling mechanism. In this application can switch from user mode and gain control of processor without involving system scheduler. It is used for applications that require high performance to execute threads concurrently. This service is available only for 64-bit versions. Here the scheduler component must be implemented by the application

1.6.3 Interprocess Communication

Interprocess communication is the mechanism which facilitates communication and sharing of data among processes or applications. Here applications requesting for a service can be called client and the application serving the request can be called server. Also applications can act as both client and server. The various IPC mechanisms available in windows are:

i. Clipboard

It is a loosely coupled medium for sharing of data. All the application needs to agree on common data format. Whenever a cut or copy operation is performed on data, it is put on clipboard which can then be used by other applications.

ii. COM

Software that uses component object model (COM) can access and communicate with variety of components. It allows one application to be embedded or linked with other components. Applications that use OLE make use of this.

iii. Data Copy

This allows sending of information from one application to another using windows messaging. The receiver must be able to identify the sender and the format of information.

iv. DDE

Dynamic data exchange is a protocol that allows variety of data formats to be exchanged. It is extension of clipboard. Unlike clipboard it continues to function without need of further interaction after one time data exchange.

v. File Mapping

In this, file is treated as block of memory that can be shared by processes. Processes that want to share this file gets a pointer to its memory location

through which they can access or even modify file contents. Processes must provide a synchronization mechanism to prevent corruption of data.

vi. Mailslots

Mailslots are used to send messages (one way) over network. Process that creates mailslot is called mail slot server. The client writes or appends the message in maislot and sends it to mail slot server. For two way communication multiple mailslots can be created. They can also be used to broadcast message across computers in network domain.

vii. Pipes

Two types of pipes can be used – anonymous or named.

- Anonymous pipes are generally used for transfer of data between related processes like parent child. Read write pipes should be created by each of the processes to share data.
- Named pipes are used between processes on different systems or to transfer among unrelated processes. Server creates pipes and communicates it to clients. Data can be exchanged once both get connected to pipe.

viii. RPC

Remote procedure call is a mechanism used to call remote functions which can be between processes of same or different machine. It also allows data to be converted for different hardware architectures. It is used for high performance applications that are tightly coupled.

ix. Windows Sockets

It is an interface used for communication between sockets. It enables data to be transmitted independent of network protocol and hence it is called protocol independent.

1.7 MEMORY MANAGEMENT IN WINDOWS 10

For efficient management of tasks in a system, proper utilization of memory is necessary. Hence memory management is one of the prime tasks performed by an operating system. The system is composed of two types of memory– Physical and Virtual memory.

(i) Physical Memory

Physical memory is also known as RAM. All the programs and data during execution along with the kernel of operating system are stored in RAM. Elements stored in this memory are directly accessible by the processor. Addresses that belong to physical memory are called physical address space. It can be further divided into user address space where user's data or program can be stored and kernel's address space where kernel is stored.

(ii) Virtual Memory

Virtual memory is capability of using hard disk as additional memory when RAM does not have sufficient capacity to store the data or programs. Data or programs stored in RAM are also mapped to virtual address space by operating system. Operating system

has a program called virtual memory manager which uses method of paging to map virtual to physical address space. If physical memory is able to accommodate the processes, then virtual addresses are directly mapped to physical address but if physical memory is not able to store processes together then virtual memory manager allocate memory to processes one by one till all the processes complete using technique disc paging and **demand Paging**.

Disc paging is space on the hard disk also called page file which is used as extension to RAM. When RAM has not enough space to keep a process it is placed in this location. However, retrieving process from this location degrades the performance. In Demand paging only the task which are currently needed, is placed in RAM.

1.7.1 Memory Representation

Physical memory used by a process is called **working set**. For each process this working set is composed of – private and sharable working sets. Private working set is the memory used by a process for dedicated tasks. Sharable working set is the memory used by tasks which are shared with tasks of other processes. All the system resources like drivers, DLL are shared resources whereas all the tasks performed through executable processes are private working set with sharable resources.

Commit memory for a process is the amount of memory reserved by operating system which is usually equal to the page file size required by a process. This memory is not allocated until it is necessary to page out a process's private working set from RAM to page file. Virtual memory required by the system from page file is hence equal to the sum of all process's commit memory. However windows allows user to modify page file and virtual memory size.

During bootup process, operating system creates two dynamic pools in physical memory for kernel components. These pools are paged and non-paged pools.

- **Paged pool** –this is the physical memory allocated for kernel components that can be later written to disk when not required.
- **Non-paged pool** –this is the physical memory allocated for kernel components or objects that should always remain in physical memory compulsorily but can optionally stored in disk.
- Allocations to both these pools can be viewed in task manager.

1.7.2 Physical memory organization

Physical memory or RAM is Windows 10 is composed of following sections:

- *Hardware reserved* - it is the memory not handled by memory manager and is locked. It is reserved by hardware drivers.
- *In use* – it is the sum of working sets of all running processes, non-paged kernel pool, drivers and applications.
- *Modified* –modified pages which remains idle for long time are removed from working set and shifted to this memory. If the size of this memory goes beyond the threshold limit, memory manager removes pages from this memory and writes it to page file.

- *Free* – it is the memory that can be allocated when demand for memory arises. Also the previously allocated memory which is no longer required are also returned to this memory by memory manager is available for reuse.
- *Standby* – this memory acts as a cache used to hold recent files. Unmodified files from working set of recently completed process are shifted to this memory by memory manager. This improves the performance of system by reducing the time required to fetch these pages if they will be required again by the system in near future.
- When a page is requested by a process, memory manager checks the availability of page in standby memory. If found, it is repurposed i.e. returned as working set. If not found in standby, the page is loaded from the hard disk to free memory and used as working set. If sufficient memory required by a process is not available in free memory than standby memory can be used for memory allocations. Priority is associated with each page in standby and the pages with low priority can be removed and allocated to needy process.

1.7.3 Virtual Memory organization

In windows 10, virtual address space is limited by the system architecture. Increasing memory beyond the limit will not improve performance. It also disallows processes to directly access primary memory and hence they are unaware of memory used by other processes. This feature provides security to the system.

Memory request made by processes are handled by Virtual Memory Manager (VMM). It allocates virtual memory in units called Page. Pages are initialized by location of hard disk where file in stored initially which is later changed to physical memory address where the file gets placed for execution. Hence virtual address can be allocated to all the processes irrespective to physical memory size. Page frames are memory units in physical memory used to store processes. Size of page frames is same as that of pages in virtual memory but they are distributed randomly across memory. Set of page frames belonging to a process forms a working set. Page tables are maintained by operating system for each process which maps pages in virtual memory to frames in physical memory if available otherwise store disk address. It is an array of addresses where index stores the virtual address and page table entries stores frame address. It also stores process control information having control bits like – Valid bit, modified bit and protection bit, which tells about the state of process.

For 32-bit architecture, maximum addressable units can be 2^{32}-1 i.e. 4GB. This is independent of physical RAM installed in system. If the RAM installed is more than 4GB then extra RAM will is added to Physical Address Extension (PAE). They are 36 bits long hence are capable of addressing 64GB physical memory. Since the instructions are created specific to architecture, process must run in 32 bit virtual memory mode for 32-bit system. Operating system will read and write from lowest 4GB RAM in 32 bit mode called active region and the region outside 32 bit is called passive which is usable in 36 bit mode. Processes are relocated from active to passive region by operating system when it ends by changing physical page number (PPN). Therefore adding more RAM only adds the capacity to cache more data which can lead to reduced disk paging.

For 64- bit architecture, maximum addressable units can be 2^{64}-1 which forms 16 Exabyte address space. Hence it allows more processes to be executed without the need of disk paging.

Memory management concepts like demand paging can be applied to both the systems since it is independent of architecture and RAM installed in system.

1.7.4 Demand Paging

For each process a process control block (PCB) is maintained by operating system kernel. In physical memory, the starting address of a page table is called Page table base address. For storing the address of page table, CPU maintains a special register called **Page table base register (PTBR)**. Process control block does the mapping between process id and page table base address. Whenever switching between processes is needed, operating system simple changes the page table base address in page table base register.

CPU maintains a register called Program Counter (PC) which is used to hold the address of next instruction to be executed in a process. Address stored in PC register is divided into two parts- higher order bits stores virtual Page number (VPN) and lower order bits store offset of element within a page. Memory management unit (MMU) first checks the address of page table from page table base register and then uses VPN of PC register as index into page table to find corresponding physical page number (PPN). PPN once found is concatenated with the offset part of PC register and the resulting address is stored in memory address register (MAR). CPU can then fetch the next instruction from the address stored in MAR and execute it. Later on the offset part of PC register is incremented by one to point to the next location.

A page table is created by operating system by mapping virtual address of processes to physical address. Pages that are required to be loaded in memory at initial stage are marked by valid bit=1. Pages that are not required initially are marked with valid bit=0. Page frames are created for initially required pages in physical memory contents are loaded into them from disk. Physical address of pages in page table is then replaced by the page frame address (PPN –physical page number). The process of bringing page into page frame or physical memory only when it is required or demanded is called **demand paging**. MMU maps virtual address to physical address only when the valid bit =1. If MMU does not find a frame number corresponding to a virtual address than Page fault is generated. This mean that the page is not loaded from the disk yet. In this case page handler is invoked to check the existence of the desired page and the page if exists is loaded from the disk into a new page frame. The frame address is then updated into the page table entry and the page is marked as valid.

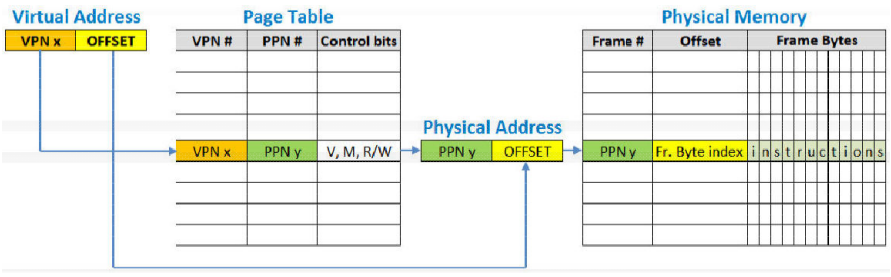

Fig 3: Mapping Virtual address to Physical address (Source: answers.microsoft.com)

1.7.5 Memory Compression

Windows 10 memory manager performs memory compression by compressing infrequently used memory pages instead of writing them on disk. This reduces number of reads and writes and improves response time. Also compressing memory makes

room for more processes to be kept in memory and hence allowing more tasks to be executed simultaneously. As compared to previous versions windows 10 writes to disk only 50 % of the time. Windows 10 also performs simultaneous decompression of data in parallel to reading data by using multiple CPU cores. This happens because of reduced number of reads and writes. The amount of memory compressed by the system can be checked in task manager.

1.8 FILE SYSTEM MANAGEMENT IN WINDOWS 10

File system enable data or files to be stored and retrived from storage locations. Naming conventions, formats for paths are also specified by file system. Storage components in windows file system are – files, directories and volumes. A file is a group of related data. A directory is collection of files and directories stored in hierarchy. A volume is also a collection of files and directories.

1.8.1 Directory Management

The number of files in a directory is limited by the size of the disk where directory is stored. The link between the directory and files within directory is implemented as **directory entry table** by NTFS file system. One or more entries are created in this table for each file in directory. Additional entries created for a file (more than one) are called hard links. Any number of hard links can be created for a file. Whenever a file is created, its entry is added into table. Similarly when a file is moved to directory, its entry is created as moved file. When a file is deleted, its entry is also deleted from table.

1.8.2 Disk Management

A hard is divided into one or more partitions, which are logical regions and created when user formats hard disk. Multiple partitions makes file management easy. In storage types there are two types of disk- basic and dynamic disk.

(i) Basic disk : Basic disk contains primary partition and logical drives. It supports Master Boot Record partition style (like older windows versions) and GUID partition table. In the traditional MBR scheme, four primary partitions or three primary and one extended can be created. Also one or more logical drives can be created in extended partition. The extended partitions should be contiguous. The partitions are formatted to create volume. One or more volumes can be created with a single partition. Partitions which are not formatted cannot be used for storage.

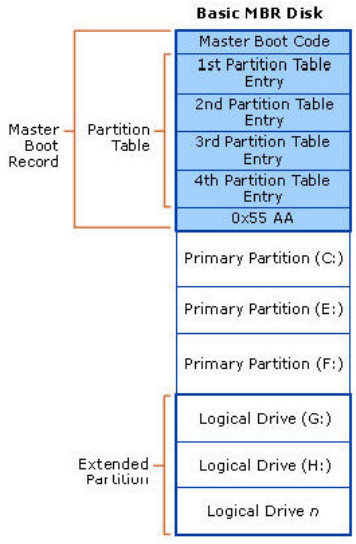

Fig4: Basic MBR Disk (Source: docs.microsoft.com)

Globally unique identifier (GUI) can be used with MBR Partition. This style allows upto 128 primary partitions to be created hence extended partitions are not needed. It allows more than 2TB partitions to be created and also applied CRC protection and replication to partitions. It is used along with MBR for backward compatibility.

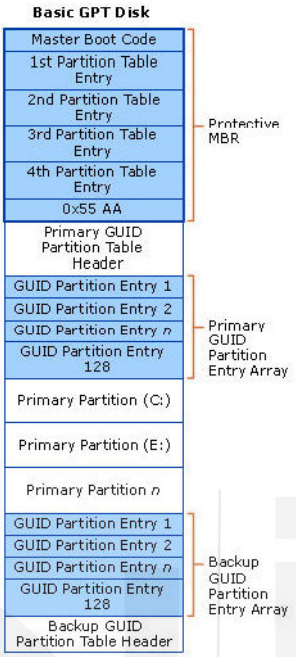

Fig 5: Basic GPT disk (Source: *docs.microsoft.com*)

(i)) **Dynamic Disks:** Similar to basic disk it can use MBR and GPT partitioning but it allows added features like- it allows volume to be spanned across multiple disks and create fault tolerant volumes. Volumes created in dynamic disk are called dynamic volumes. Information of dynamic volumes is maintained in a database. It allows volume to be created on one or more physical disk noncontiguously. It makes use for logical Disk manager (LDM) and Virtual Disk service (VDS). For converting basic disk to dynamic disk atleast 1 MB of unused space is required in disk for LDM database requirements.

1.8.3 File Management

In windows 10, all directories, files, system code are stored by NTFS file system in a file. This help in easy access and maintenance and can also be protected by security descriptor unlike other file system which stores this data in regions of disk external to file system. Group of sectors called a cluster is the fundamental storage unit of file system. The disk data can be administered by disk controller independent of the sector size.

Files are managed in windows through file objects, file pointers and file handles.

1. A file object maintains contents written to file and other attributes maintained by kernel like- file name, current offset, share mode and others. Hence it creates a logical interface between file data and process of kernel and user mode.

2. File handle is associated with a file when a process opens the file. It remains associated with the file until the process ends or the handle is closed. They are unique and private to a process.

3. File pointer is an offset value that specifies the next byte to be read or written from a file. It is created by the system and placed at the beginning of file (offset 0) when it is opened. It progresses through the file with number of reads and writes.

1.8.4 File compression

Compression of file is supported by NTFS file system. It uses a lossless compression algorithm called Lempel-Ziv compression to compress files. This allows data to be decompressed without any loss of information. Compression of files larger than 30 GB may not be successful. Applications cannot access compressed data but can only operate on those files with assistance of file compression library .

☞ **Check Your Progress 1**

1) List the important new features of WINDOWS 10 Operating System.

.....

.....

.....

.....

2) Describe the security features in Windows 10.

.....

.....

.....

.....

1.9 SUMMARY

In this unit a case study of Windows 10 is presented. The various operating system functions like- Process management, memory management and file System management are discussed in details. Also various features newly added in Windows 10 operating system and various available versions introduced are also mentioned. In this unit, we also discussed several theoretical concepts on Windows 10 in detail, often useful in your lab for practice.

1.10 SOLUTIONS/ANSWERS

1) Some of the new features and enhancements that will allow you to benefit from WINDOWS 10 are intelligent security, simplified updates, flexible management, and enhanced productivity. Apart from these following are the new features in WINDOWS 10:

   - Theme aware tiles in Start
   - Microsoft Edge
   - Improved Notifications

- Tablet Experience
- Refresh Rate of Display
- Mobile Device Management
- Windows Autopilot (for HoloLens, Co-Management etc.)
- Defender Application
- Latest Cumulative Updates (LCUs) and Servicing Stack Updates(SSUs) combined into a single cumulative monthly update via Microsoft catalog
- Secure Biometric Sign-on
- Cortana
- Universal Print
- Virtual Desktop
- MS Tunnel Gateway
- EndpointAnalysis
- Microsoft 365 Apps
- Productivity Score

2) Windows 10 includes Windows Security, which provides the latest antivirus protection. Your device will be actively protected from the moment you start Windows 10.

- Windows Security continually scans for malware (malicious software), viruses, and security threats. In addition to this real-time protection, updates are downloaded automatically to help keep your device safe and protect it from threats.
- Windows Security is built-in to Windows 10 and includes an antirvirus program called Microsoft Defender Antivirus.
- Virus & threat protection: Monitor threats to your device, run scans, and get updates to help detect the latest threats. (Some of these options are unavailable if you're running Windows 10 in S mode.)
- Account protection: Access sign-in options and account settings, including Windows Hello and dynamic lock.
- Firewall and network protection: Manage firewall settings and monitor what's happening with your networks and internet connections.
- App and browser control: Update settings for Microsoft Defender SmartScreen to help protect your device against potentially dangerous apps, files, sites, and downloads. You'll have exploit protection and you can customize protection settings for your devices.
- Device security. Review built-in security options to help protect your device from attacks by malicious software.

- Device performance and health: View status info about your device's performance health, and keep your device clean and up to date with the latest version of Windows 10.
- Kids Online Activity Protection: Keep track of your kids' online activity and the devices in your household.

1.11 FURTHERREADINGS

1. Andy Mince, WINDOWS 10 Made Simple, Kindle Edition, 2020.
2. Katherine Murray, My Windows 10, Second Edition, Que, 2018.
3. Andy Rathbone, Windows 10 for Dummies, 4th Edition, Wiley, 2020.
4. docs.microsoft.com

UNIT 2 CASE STUDY - LINUX

Structure

2.1 INTRODUCTION

Operating system is system software, which acts as an interface between user applications and hardware devices such as input/output devices, memory, file system, etc. It schedules both type of tasks, systems and users', and provides common core services such as a basic user interface. The two major goals of any operating system (OS) are:

- **Convenience:** It transforms the raw hardware into a machine that is more accessible to users.
- **Efficiency:** It efficiently manages the resources of the overall computer system.

Operating system consists of the following components for achieving the goals:

- **Kernel:** The main task of the kernel is to optimize the usage of the hardware, running the required programs, satisfying the user's requirements.
- **Application Programming Interface (API):** A collection of rules, which describes how services have to be required from the kernel, and how we receive the answer from it.
- **Shell**: Its' task is the interpretation of commands. The shell can be command line (CLI - Command Line Interface, such as DOS), or graphic - GUI - interface (e.g.: Windows)
- **Services and Utilities:** These supplementary programs enhance the user's experience (e.g.: word processors, translator programs) and are not inseparable parts of the system.

Operating systems may be categorized into various types based on their functionality such as single user single task operating systems, single user multitasking operating systems, multi-programmed operating system, multiuser operating systems, distributed operating systems, network operating system, multiprocessor operating system, real time operating systems and embedded operating systems. Disk Operating System (DOS), Windows, UNIX, LINUX, Macintosh (macOS) are some of the examples of the operating systems. The typical services that an operating system provides include: a task scheduler, memory manager, disk manager, network manager, Other I/O services and security manager. This section provides a case study of LINUX operating system. This unit provides basic structure of the Linux, its process management; file management and memory management techniques.

2.2 OBJECTIVES

After the completion of this unit, you will be able to:

- Understand the basic functions of Linux Operating System
- Identify the characteristics of the Linux Operating System
- Know the evolution of Linux operating system
- Understand the process management of Linux and compare with other OS
- Understand the memory management approaches in Linux
- Understand the File management in Linux
- Understand the security features of Linux

2.3 LINUX OPERATING SYSTEM

UNIX is a well-known OS, and is widely used in mini and mainframe computer systems. In recent years the popularity of UNIX has been increasing significantly due to one of its popular descendant namely Linux operating system.

Linux is one of the popular variants' of UNIX operating System. It is an open source OS as its source code is freely available. Linux was designed considering UNIX compatibility. Its functionality is quite similar to that of UNIX. Linux is provided with GUI, so instead of typing commands to perform user tasks, one can log in graphically to fulfill their requirements. Linux also provides the facility to handle core system requirements through command line. Linux developers concentrated on networking and services in the beginning, and office applications have been the last barrier to be taken down. Apart from desktop OS, Linux is an acceptable choice as a workstation OS, providing an easy user interface and MS compatible office applications like word processors, spreadsheets, presentations and the like. Because of this, Linux has joined the desktop market.

On the server side, Linux is known as a stable and reliable platform, providing database and trading services for companies like Amazon, US Post Office, German Army and many others. Linux is used in firewall, proxy and web server. One can find a Linux box within reach of every UNIX system administrator who appreciates a comfortable management station. Clusters of Linux machines are used in the creation of movies

such as "Titanic", "Shrek" and others. Day-to-day, thousands of heavy-duty jobs are performed by Linux across the world. It is also noteworthy that modern Linux not only runs on workstations, mid and high-end servers, but also on "gadgets" like PDA's, mobiles, embedded applications and even on experimental wristwatches. This makes Linux the only operating system all across covering a wide range of hardware.

Linux is used in a number of consumer electronic devices worldwide. Some of the popular Linux based electronic devices are Dell Inspiron Mini (9 and 12), Garmin Nuvi (860, 880, and 5000), Google Android Dev Phone 1, HP Mini 1000, Lenovo IdeaPad S9, Motorola MotoRokr EM35 Phone, One Laptop Per Child XO2, Sony Bravia Television, Sony Reader, TiVo Digital Video Recorder, Volvo In-Car Navigation System, Yamaha Motif Keyboard etc.. From smartphones to robots, cars, supercomputers, home appliances, personal computers to Enterprise S ervers, the Linux operating system is everywhere.

2.4 EVOLUTION OF LINUX

By the beginning of the 90s home PCs were finally powerful enough to run UNIX. Linus Torvalds, a Computer Science student at the University of Helsinki, thought it would be a good idea to have some sort of freely available academic version of UNIX, and promptly started to code. He started to put fourth questions, looking for answers and solutions that would help him get UNIX on his PC. From the start, it was Linus's goal to have a free system that was completely compliant with the original UNIX. In those days plug-and-play wasn't invented yet, but number of people was interested in having a UNIX system of their own. This was the only small obstacle. New drivers became available for all kinds of new hardware at a continuous rising speed. As soon as a new piece of hardware became available, someone bought it and submitted it to the Linux test, as the system was gradually being called releasing more free code for an even wider range of hardware. Around 95% of the Linux was written in C programming language and around 2.8% in *Assembly* language.

Two years after Linus' announcement of the project, there were 12000 Linux users. The project, popular with hobbyists, grew steadily, all the while staying within the bounds of the POSIX (popular UNIX version) standard. All the features of UNIX were added over the next couple of years, resulting in the mature operating system Linux has become today. Linux is a full UNIX clone, fit for use on workstations as well as on middle-range and high-end servers. There is a wide range of variations and versions those were encountered in the development of Linux Operating System. The following table provides the details of the various versions developed and events happened related to Linux. The history of Linux is given in Table1.

Table 1: Evolution of Linux

S.No.	Year	Event/Release	Version
1	1991	UniX(HP-UX)	8.0
2	1992	Hewlett Packard	9.0
3	1993	NetBSD and FreeBSD	0.8, 1.0
4	1994	Red Hat Linux, Caldera, Ransom Love and NetBSD1.0	—

5	1995	FreeBSD and HP UX	2.0, 10.0
6	1996	K Desktop Environment	—
7	1997	HP-UX	11.0
8	1998	IRIXSun Solaris OS Free BSD	6.573.0
9	2000	Caldera Systems with SCO server division announced	
10	2001	LinuxMicrosoft filed a trademark suit against Lindows.com	2.4
11	2004	Lindows name was changed to Linspire First release of Ubuntu	—
12	2005	openSUSE Project	—
13	2006	Red Hat	—
14	2007	Dell started distributing laptops with Ubuntu pre-installed	
16	2011	Linux kernel	3.0
17	2013	Googles Linux based Android	—

Linux has several distributions. Some popular and dominant LINUX distributions for Desktops include:

- LinuxMint
- Ubuntu
- OpenSUSE
- Mageia
- Manjaro
- Fedora
- Arch Linux
- Debian
- Kali Linux
- Elementary OS

Some of the popular Linux distributions, in the Servers category, include:

- Red Hat Enterprise Linux (RHEL)
- CentOS
- Ubuntu Server
- SUSE Enterprise Linux

Let us see the architecture of LINUX in the next section.

2.5 CHARACTERISTICS OF LINUX

Linux has several salient characteristic features, some of the important among them are:

- **Multiuser Capability:** This is a capability of Linux OS where, the same computer resources – hard disk, memory, etc. are accessible to multiple users. Of course, not on a single terminal, they are given different terminals to operate from. Aterminal will consist of at least a Monitor/VDU, keyboard and mouse as input devices. All the terminals are then connected to the main Linux Server or Host Machine, whose resources and connected peripheral devices such as printer, can be used. Client/Server Architecture is an example of multiuser capability of Linux, where different clients are connected to a Linux server. The client sends request to the server with a particular data and server requests with the processed data or the file requested, client terminal is also known as a Dumb Terminal.
- **Multitasking:** Linux has the ability to handle more than one job at a time, say for example you have executed a command for sorting for a huge list and simultaneously typing in a notepad. This is managed by dividing the CPU time intelligently by the implementation of scheduling policies and the concept of context switching.
- **Portability***:* Portability was the one of the main features that made Linux so popular among the users, but portability doesn't mean that it is smaller in file size and can be carried on pen drive, CDs and memory cards. Instead, here portability means that Linux OS and its application can work on different types of hardwares in the same way. Linux kernel and application programs support their installation even on very least hardware configuration.
- **Security:** Security is a very important part of any OS, for the organizations/ user who is using the system for their confidential works, Linux does provide several security concepts for protecting their users from unauthorized access of their data and system.

Linux provide three main security concepts:

- **Authentication:** This simply implies claiming the person whom you are by assigning passwords and login names to individual users, ensuring that nobody can gain access to their work.
- **Authorization:** At the file level, Linux has authorization limits to users. There are read, write and execute permissions for each file which decide who can access a particular file, who can modify it and who can execute it.
- **Encryption:** This feature encodes your files into an unreadable format that is also known as "cyphertext", so that even if someone succeeds in opening it your secrets will be safe.

(i) Communication*:* Linux has an excellent feature for communicating with the fellow users; it can be within the network of a single main computer, or between two or more computer networks. The users can easily exchange mail, data, program through such networks.

Advantages of LINUX Operating System

There are several advantages of LINUX operating system. Some of them are as follows:

- **Low cost:** There is no need to spend time and huge amount money to obtain licenses since Linux and much of its software come with the GNU General Public License. There is no need to worry about any software's that you use in Linux.
- **Stability:** Linux has high stability compared with other operating systems. There is no need to reboot the Linux system to maintain performance levels. Rarely it freeze up or slow down. It has a continuous up-time of hundreds of days or more.
- **Performance:** Linux provides high performance on various networks. It has the ability to handle large numbers of users simultaneously.
- **Networking:** Linux provides a strong support for network functionality; client and server systems can be easily set up on any computer running Linux. It can perform tasks like network backup faster than other operating systems.
- **Flexibility:** Linux is very flexible. Linux can be used for high performance server applications, desktop applications, and embedded systems. You can install only the needed components for a particular use. You can also restrict the use of specific computers.
- **Compatibility:** It runs all common Unix software packages and can process all common file formats.
- **Wider Choice:** There is a large number of Linux distributions which gives you a wider choice. Each organization develop and support different distribution. You can pick the one you like best; the core functions are the same.
- **Fast and easy installation:** Linux distributions come with user-friendly installation.
- **Better use of hard disk:** Linux uses its resources well enough even when the hard disk is almost full.
- **Multitasking:** Linux is a multitasking operating system. It can handle many things at the same time.
- **Security:** Linux is one of the most secure operating systems. File ownership and permissions make Linux more secure.
- **Open source:** Linux is an Open source operating systems. You can easily get the source code for Linux and edit it to develop your personal operating system.

2.6 LINUX KERNEL

The Linux kernel is the main component of a Linux OS. This is the core interface between a computer's hardware and its processes. It communicates between the two, managing resources as efficiently as possible. The kernel is so named because—like a

seed inside a hard shell—it exists within the OS and controls all major functions of the hardware, whether it's a phone, laptop, server, or any other kind of computer. To put the kernel in context, one can think of a Linux system having four layers as shown in Fig 1.

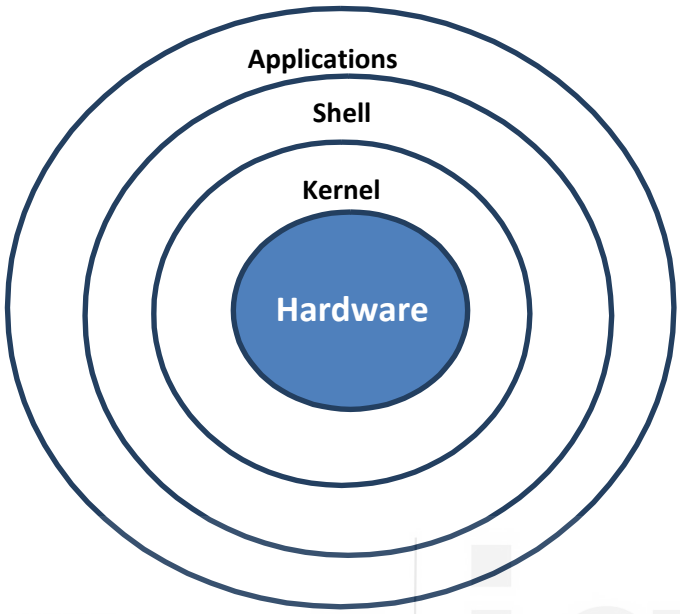

Fig 1: Linux Architecture

1. **The Hardware:** The physical machine—the bottom or base of the system, made up of memory (RAM), processor or central processing unit (CPU), as well as input/output (I/O) devices such as storage, networking, and graphics. CPU performs computations and reads from, and writes to the memory.
2. **The Linux kernel:** The core of the OS. It's software residing in memory that tells the CPU what to do.
3. **Shell:** The shell layer is in between application layer and the kernel. Shell will take the input from the user applications and sends it to the kernel in form of instructions. It also takes output from the kernel and forwards it as a result to the user application.
4. **Applications / User processes:** These are the running programs that the kernel manages. User processes are what collectively make up user space. User processes are also known as just *processes*. The kernel also allows these processes and servers to communicate with each other (known as inter-process communication, or IPC).

Code executed by the system runs on CPUs in one or two modes: kernel mode or user mode. Code running in the kernel mode has unrestricted access to the hardware, while user mode restricts access to the CPU and memory to the System Call Interface. A similar separation exists for memory (kernel space and user space).

2.5.1 Kernel Architecture of Linux

Kernel is a small and special code which is the core component of Linux OS and directly interacts with hardware. It is the intermediate level between software and hardware which provides low level service to user mode's components. It is developed in C language. Moreover, it has different blocks which manage various operations. Kernel runs a number of processes concurrently and manages various resources. It is

viewed as a resource manager when several programs run concurrently on a system. In this case, the kernel is an instance that shares available resources like CPU time, disk space, network connections etc.

The kernel has four jobs to perform as follows:

i. **Memory management:** Keep track of how much memory is used to store what, and where.

ii. **Process management:** Determine which processes can use the central processing unit (CPU), when, and for how long.

iii. **Device drivers:** Act as mediator/interpreter between the hardware and processes.

iv. **System calls and security:** Receive requests for service from the processes

The kernel, if implemented properly, is invisible to the user, working in its own little world known as kernel space, where it allocates memory and keeps track of where everything is stored. What the user sees—like web browsers and files—are known as the user space. These applications interact with the kernel through a system call interface (SCI).

This also means that if a process fails in user mode, the damage is limited and can be recovered by the kernel. However, because of its access to memory and the processor, a kernel process crash can crash the entire system. Since there are safeguards in place and permissions required to cross boundaries, user process crashes usually can't cause too many problems.

2.6 FUNDAMENTAL ARCHITECTURE OF LINUX

The following is the fundamental architecture of Linux given in Fig 2.

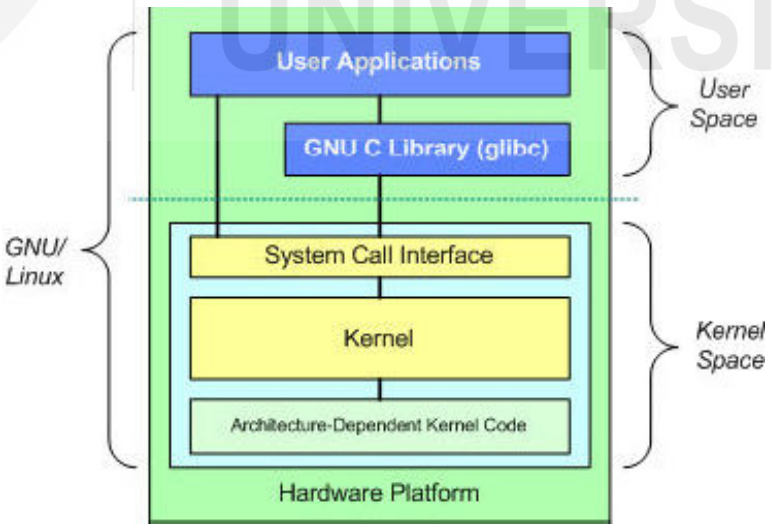

Fig. 2: Fundamental Architecture of Linux

Architecture of kernel is divided into main two parts:

1. User Space
2. Kernel Space

2.6.1 User Space

All user programs and applications are executed in user space. User Space cannot directly access the memory and hardware. It accesses the hardware through kernel space. Processes or programs which are running in user space only access some part of memory by system call. Due to full protection, crashes in user mode are recoverable.

GNU C library provides the mechanism switching user space application to kernel space.

2.6.2 Kernel Space

All kernel programs are executed in kernel space. Kernel space accesses full part of memory and directly interacts with hardware like RAM, Hard disk etc. It is divided in different blocks and modules which manage all operations (like file management, memory management, process management etc.) in kernel space and applications running in user space. Kernel space consists of system call interface, Kernel (core component of Linux) and device module.

System call interface is the intermediate layer between user space and kernel space. Each application, running in user space, can interface with kernel through system call. For example system call function on file operation are open( ), write( ), read( ) etc.

Kernel is independent from hardware. It is common for all Hardware processors which are supported by Linux. You can run kernel on any processor like Intel, ARM, etc. It acts as a resource manager in Kernel space and performs process management, file management, memory management, Interrupt handler, scheduling of process, etc. It is a powerful structure which handles all kinds of operations.

2.7 PROCESS MANAGEMENT IN LINUX

Multitasking is the illusion created by the kernel. The processor is capable of performing the processes in parallel by switching in between multiple tasks that are running on the system. The context switching is done in parallel rapidly and repeatedly in specific intervals. The user is not able to notice the switching of the tasks due to small intervals. Linux process can be visualized as running instance of a program. For example, just open a text editor on your Linux box and a text editor process will be born.

The following are some of the tasks of process management that are handled by the kernel.

All the processes should work independently without interfering each other until they desired to interact. The Linux environment is a multiuser system. Even though multiple processes are running concurrently, the problem occurred in one application will not affect some other application. All these applications/programs that are running in parallel will not be able to access the memory content of the other application/program. This feature provides the security for the application's data from the other users.

All the processes of the system will fairly utilize CPU time and share in between multiple applications. The applications are assigned their priorities based on which the order of the execution is determined.

The execution time allocated to each process (time quantum) and when to context switch between processes should take place is decided by the kernel. This begs the

question as to which process is actually the next. Decisions on time slot allocation and context switching are not platform-dependent.

While context switching by the kernel, the kernel will ensure that the execution environment of a process brought back is exactly the same when it last withdrew processor resources. For example, the contents of the processor registers and the structure of virtual address space must be identical. This latter task is extremely dependent on processor type. How CPU time is allocated is determined by the scheduler policy which is totally separate from the task switching mechanism needed to switch between processes.

Process Priorities: Not all processes are of equal importance. In addition to process priority, there are different criticality classes to satisfy differing demands. In general processes can be split into real-time processes and non-real-time processes.

Real-time processes are also of two types i.e., hard real-time and delicate real-time or soft real-time processes. Hard real-time processes are subject to strict time limits during which certain tasks must be completed. If the flight control commands of an aircraft are processed by computer, they must be forwarded as quickly as possible — within a guaranteed period of time. The key characteristic of hard real-time processes is that they must be processed within a guaranteed time frame. Note that this does not imply that the time frame is particularly short. Instead, the system must guarantee that a certain time frame is never exceeded, even when unlikely or adverse conditions prevail.

In soft real time systems, the meeting of deadline is not compulsory for every time for every task but process should get processed and give the result. Even the soft real time systems cannot miss the deadline for every task or process according to the priority it should meet the deadline or can miss the deadline. If system is missing the deadline for every time the performance of the system will be worse and cannot be used by the users. Best example for soft real time system is personal computer, audio and video systems, etc.. An illustration of a delicate real-time process is a compose activity to a CD. Information should be procured by the CD author at a specific rate since information are kept in touch with the medium in a nonstop stream. On the off chance that framework stacking is too high, the information stream might be intruded on quickly, and this may bring about an unusable CD, far less exceptional than a plane accident. All things considered, the compose process should consistently be conceded CPU time when required — before any remaining typical processes.

Linux does not support hard real-time processing, at least not in the vanilla kernel. There are, however, modified versions such as RTLinux, Xenomai, or RATI that offer this feature. The Linux kernel runs as a separate ‘‘process’’ in these approaches and handles less important software, while real-time work is done outside the kernel. The kernel may run only if no real-time critical actions are performed. Since Linux is optimized for throughput and tries to handle common cases as fast as possible, guaranteed response times are very hard to achieve. Nevertheless quite a bit of progress has been made during last years to decrease the overall kernel latency, i.e., the time that elapses between making a request and its fulfilment.

Process Hierarchy

Each process is identified by a unique positive integer called the process ID (pid). The *pid* of the first process is 1, and each subsequent process receives a new, unique *pid*.

In Linux, processes form a strict hierarchy, known as the process tree. The process tree is rooted at the first process, known as the init process, which is typically the init program. New processes are created via the fork() system call. This system call creates a duplicate of the calling process. The original process is called the parent; the new process is called the child. Every process except the first has a parent. If a parent process terminates before its child, the kernel will reparent the child to the init process.

When a process terminates, it is not immediately removed from the system. Instead, the kernel keeps parts of the process resident in memory to allow the process's parent to inquire about its status upon terminating. This inquiry is known as waiting on the terminated process. Once the parent process has waited on its terminated child, the child is fully destroyed. A process that has terminated, but has not yet been waited upon, is called a zombie. The init process routinely waits on all of its children, ensuring that re-parented processes do not remain zombies forever.

Linux Process States

Processes are born, share resources with parents for some time, get their own copy of resources when they are ready to make changes, go through various states depending upon their priority and then finally die. Following are the various states of Linux processes:

- RUNNING – This state specifies that the process is either in execution or waiting to get executed.
- INTERRUPTIBLE – This state specifies that the process is waiting to get interrupted as it is in sleep mode and waiting for some action to happen that can wake this process up. The action can be a hardware interrupt, signal etc.
- UN-INTERRUPTIBLE – It is just like the INTERRUPTIBLE state, the only difference being that a process in this state cannot be waken up by delivering a signal.
- STOPPED – This state specifies that the process has been stopped. This may happen if a signal like SIGSTOP, SIGTTIN etc is delivered to the process.
- TRACED – This state specifies that the process is being debugged. Whenever the process is stopped by debugger (to help user debug the code) the process enters this state.
- ZOMBIE – This state specifies that the process is terminated but still hanging around in kernel process table because the parent of this process has still not fetched the termination status of this process. Parent uses wait() family of functions to fetch the termination status.
- DEAD – This state specifies that the process is terminated and entry is removed from process table. This state is achieved when the parent successfully fetches the termination status as explained in ZOMBIE state.

Linux Threads Vs Light Weight Processes

Threads in Linux are nothing but a flow of execution of the process. A process containing multiple execution flows is known as multi-threaded process. For a non multi-threaded process there is only execution flow that is the main execution flow and hence it is also known as single threaded process.

Threads are often mixed with the term Light Weight Processes (LWPs). The reason dates back to those times when Linux supported threads at user level only. This means that even a multi-threaded application was viewed by kernel as a single process only. This posed big challenges for the library that managed these user level threads because it had to take care of cases that a thread execution did not hinder if any other thread issued a blocking call. Later on the implementation changed and processes were attached to each thread so that kernel can take care of them. Linux kernel does not see them as threads; each thread is viewed as a process inside kernel. These processes are known as light weight processes.

The main difference between a LWP and a normal process is that LWPs share same address space and other resources like open files etc. As some resources are shared so these processes are considered to be light weight as compared to other normal processes and hence the name light weight processes.

So, effectively we can say that threads and light weight processes are same. It's just that thread is a term that is used at user level while light weight process is a term used at kernel level.

From implementation point of view, threads are created using functions exposed by POSIX compliant pthread library in Linux. Internally, the clone() function is used to create a normal as well as a light weight process. This means that to create a normal process fork() is used that further calls clone() with appropriate arguments while to create a thread or LWP, a function from pthread library calls clone() with relevant flags. So, the main difference is generated by using different flags that can be passed to clone() function.

2.8 MEMORY MANAGEMENT IN LINUX

The major part of the computer is CPU of which RAM is the frontal part of the CPU. All data items processed at CPU has to go through RAM. A process, which needs to be processed, will first be loaded in RAM and the CPU will get process data from RAM.

Memory is assisted by level one, level two, and level three cache for faster processing. As the caches are very expensive due to technology used and also not very useful for all the instructions, only small size caches are used in CPU.

Any information, related to the process, is copied from RAM to CPU registers and the CPU would build its cache. Cache plays an important part of memory management on Linux. If user requests information from hard disk, it is copied to RAM and the user is served from RAM. While the information is copied from hard disk, it is placed in page cache.

So the **page cache** stores recently requested data to make it faster if the same data is needed again. And if the user starts modifying data, it will go to RAM as well where from it will be copied to the hard However, it only happens if the data has been staying in RAM long enough.

Data won't be written immediately from RAM to hard disk, but to optimize the write operations to the hard disk, Linux works with the concept of **dirty cache**. It tries to buffer as much data in order to create an efficient write request. From figure 3, it is clear that everything on a computer goes through RAM. So using RAM on a Linux computer is essential for the well working of the Linux operating system.

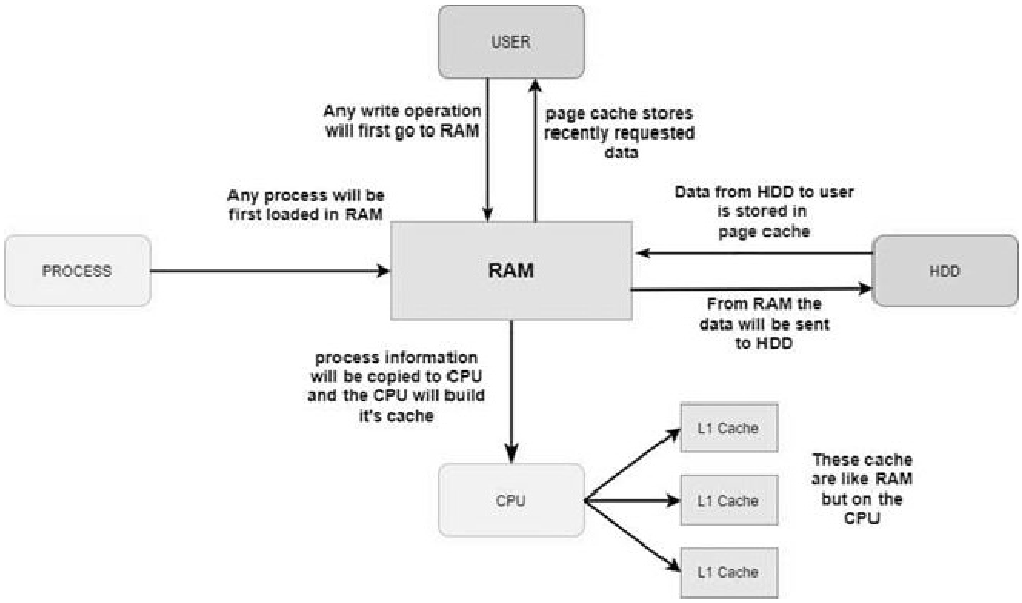

Figure 3: Memory Management in Linux

2.9 LINUX FILE SYSTEM

A file system is a logical collection of files on a partition or disk. A partition is a container for information and can span an entire hard drive if desired. File system hierarchy standard (FHS) describes directory structure and its content in Unix and Unix like operating system. It explains where files and directories should be located and what it should contain. Linux partially follows FHS due to its own policy to which we may notice some differences in the directory tree structure.

The Root Directory

All the directories in the Linux system come under the root directory which is represented by a **forward** slash **(/)**.

Each file is categorized based on the type of the file and is stored into a particular directory. These types of directories and relevant file types are listed below in table 2.

Table 2: Directory Type and Types of the Files Stored

Directory type	Types of files stored
Binary directories	Contains binary or compiled source code files, eg, /bin, /sbin, etc.
Configuration directories	Contains configuration files of the system, eg, /etc, /boot.
Data directories	Stores data files, eg, /home, /root, etc.
Memory directories	Stores device files which doesn't take up actual hard disk space, eg, /dev, /proc, /sys.
Usr (Unix System Resources)	Contains sharable, read only data, eg, /usr/bin, /usr/lib, etc.
var (variable directory)	Contains larger size data, eg, /var/log, /var/cache, etc.
Non-standard directories	Directories which do not come under standard FHS, eg, lost+found, /run, etc.

2.9.1 Linux Binary Directory

Binary files are the files which contain executable files. These files are the output of compilation of the source code. These executable files can be executed on the computer. Binary directory contains following directories:

- **/bin**
- **/sbin**
- **/lib**
- **/opt**

/bin: The '/bin' directory contains user binaries, executable files, Linux commands that are used in single user mode, and common commands that are used by all the users, like cat, cp, cd, ls, etc.

/sbin: The '/sbin' directory also contains executable files, but unlike '/bin' it only contains system binaries which require root privilege to perform certain tasks and are helpful for system maintenance purpose. e.g. fsck, root, init, ifconfig, etc.

/lib : The '/lib' directory contains shared libraries which are often used by the '/bin' and '/sbin' directories. It also contains kernel module.

/opt: The term 'opt' is short for optional. Its main purpose is to store optional application software packages. Add-on applications from individual vendors should be installed in '/opt'. And so in some systems '/opt' is empty as they may not have any add-on application.

2.9.2 Linux Configuration Directory

The configuration directory contains configured files which configures the parameters and initial settings for some computer programs.

Configuration directory have following sub-directories:

- **/boot**
- **/etc**

/boot : The '/boot' directory contains boot loader files which are essential to boot the system. In other words, they only contain files which are needed for a basic Linux system to get up and going.You may find '/boot/grub'directory which contains '/boot/grub/grub.cfg' (older system may have /boot/grub/grub.conf) which defines boot menu that is displayed before the kernel starts.

/etc : All the machine related configuration files are kept in '/etc'. Almost everything related to the configuration of your system is placed here. It also containstartup and shutdown shell script which is used to start and stop a program.All the files are static and text based and no binary files can be placed in this directory.

2.9.3 Linux Data directory

Data directory is used to store data of the system. Data directory contains following directories:

- **/home**

- **/root**
- **/srv**
- **/media**
- **/mnt**
- **/tmp**

/home: The '/home'directory stores users personnel files. After the '/home'there is a directory which is generally named at the user's name like we have '/home/sssit'. Inside this directory we have our sub-directories like Desktop, Downloads, Documents, pictures, etc.

/root: The '/root' directory is the home directory of the root user. The '/root' directory is different from (/) root.

/srv: The term 'srv' is short for **service**. The '/srv' directory contains server specific data for services provided by the system like www, cvs, rysync, ftp, etc.

/media: The '/media' directory acts as a mount point for removable media devices such as CD-Rom, floppy, USB devices, etc.This is newly introduced directory and hence a system can run without this directory also.

/mnt : The term 'mnt' stands for **mount**. The '/mnt' directory should be empty and sysadmins can only mount temporary filesystems.

/tmp : The term 'tmp'stands for **temporary**. Data stored in '/tmp' is temporary and may use either disk space or RAM. When system is rebooted, files under this directory is automatically deleted. So it is advisable that never use '/tmp' to store important data.

2.9.4 Linux Memory Directory

Memory directory contains files of the whole system. All the device information, process running in data or system related information is stored in this directory. Memory directory contains the following directories:

- **/dev**
- **/proc**
- **/sys**

/dev: The term 'dev'is short for **device**. As you know in Linux operating system everything is a file. It appears to be an ordinary file but doesn't take up disk space. Files which are used to represent and access devices are stored here including terminal devices like usb. All the files stored in '/dev' are not related to real devices, some are related to virtual devices also.

/dev/ tty and /dev/ pts: The '/dev/tty' file represents the command line interface that is a terminal or console attached to the system. Typing commands in a terminal is a part of the graphical interface like Gnome or KDE, then terminal will be represented as '/dev/pts/1'.

/dev/ null: The '/dev/null' file is considered as black hole, it has unlimited storage but nothing can be retrieved from it. You can discard your unwanted output from the terminal but can't retrieve it back.

/proc: The term 'proc' is short for process. Same as '/dev', '/proc' also doesn't take up disk space. It contains process information. It is a pseudo file system that contains information about running processes. It also works as virtual file system containing text information about system resources.

/sys: The term 'sys' is short for system. Basically it contains kernel information about hardware. It was created for Linux 2.6 kernel. It is a kind of '/proc' and is used for plug and play configuration.

2.9.5 Linux System Resources (/usr)

Although it is pronounced as user but in actual it stands for **Unix System Resources**. It is also called secondary hierarchy as it contains binaries, libraries, documentation for all the user applications. It only contains shareable read-only data. The following are some of the /usr sub-directories:

- **/usr/bin**
- **/usr/include**
- **/usr/lib**
- **/usr/share**
- **/usr/local**
- **/usr/src**

/usr/bin: The '/usr/bin' directory contains non-essential binary commands for all users. If you can't find a command in '/bin', search it in '/usr/bin'. It contains a lot of commands.

/usr/include: The '/usr/include' directory contains standard include files for C.

/usr/lib: The '/usr/lib' directory contains libraries that are not directly executed by the users. In other words, it contains binaries for the '/usr/bin' and '/usr/sbin'.

/usr/share: The '/usr/share' directory contains architecture independent (shared) data.

/usr/local : The '/usr/local'directory is used to install software locally. It means all the user programs that you'll install from source will be installed here.

/usr/src : The term 'src' is short for **source**. It is used to store source code like kernel source code with its header files.

2.9.6 Variable Directory (/var)

The term 'var' is short for **variable**. Files that have an unexpected size and whose content is expected to change continuously (that's why it is named as variable) during normal operation of the system are stored here. For example, log files, spool files and cache files. The following are some of the /var sub-directories here:

- **/var/log**
- **/var/cache**

- **/var/spool**
- **/var/lib**

/var/log : The '/var/log' directory contains all log files.

/var/cache: The '/var/cache' directory stores application cache data. Cache data are locally generated by I/O or calculation. Cache must be able to regenerate or restore the data. These files can be deleted without any loss of data.

/var/spool: The '/var/spool' directory is used to spool the files waiting to be processed. For example, printing queues and mail queues.

/var/lib: The '/var/lib' directory stores the files that contains state information like databases. File's data modifies as their respective programs run.

2.9.7 Non-Standard Directories

Directories which do not come under the standard FHS are called non-standard directories.Non-standard directories are as follows:

- /cdrom
- /run
- /lost + found

/cdrom: The '/cdrom' directory is not in the standard FHS but cdrom can be mounted on this directory. Ideally according to standard FHS cdrom should be mounted under '/media'.

/run: The '/run' directory stores run-time variable data. Run-time vriable data means, data about the running system since last boot. For example, running daemons.

/lost+found: During system crash or in any other situation when Linux file system checker (fsck) recovers lost data, that data is stored in this directory. Data may or may not be in a good condition.

2.10 SECURITY FEATURES IN LINUX

Linux is both more secure and less common than Windows based systems with the consequence that attacks on Linux systems occur less frequently than on Windows systems. Some of the security features of Open Source UNIX-like operating systems, focusing on Linux distributions are as follows:

2.10.1 User Accounts

Linux OS includes a root account, which is the only account that may directly carry out administrative functions. All of the other accounts on the system are *unprivileged*. This means these accounts have no rights beyond access to files marked with appropriate permissions, and the ability to launch network services. Only the root account may launch network services that use port numbers lower than 1024. Any account may start network services that use higher port numbers. Each user should have a single account on the system. Network services may also have their own separate accounts,

in order to be able to access those files on the system that they require. Utilities enable authorized users to temporarily obtain root privileges when necessary, so that administrators may manage the system with their own user accounts. For convenience, accounts may be members of one or more *groups*. If a group is assigned access to a resource, then every member of the group automatically has that access. The majority of UNIX-like systems use a Pluggable Authentication Modules (PAM) facility to manage access by users. For each login attempt or password change, the relevant service runs the configured *PAM* modules in sequence. Some modules support authentication sources, such as locally-stored files or *LDAP* directory services. Administrators may enable other modules that carry out setup tasks during the login process, or check login requests against particular criteria, such as a list of time periods when access is permitted.

2.10.2 File Permissions

Every file and directory on a Linux is marked with three sets of file permissions that determine how it may be accessed, and by whom:

- The permissions for the *owner*, the specific account that is responsible for the file
- The permissions for the *group* that may use the file
- The permissions that apply to all *other* accounts

Each set may have none or more of the following permissions on the item – *read, write* and *execute*

A user may only run a program file if they belong to a set that has the *execute* permission. For directories, the *execute* permission indicates that users in the relevant set may see the files within it, although they may not actually read, write or execute any file unless the permissions of that file permit it. Executable files with the *setUID* property automatically run with the privileges of the file owner, rather than the account that activates them. Avoid setting the execute permission or setUID on any file or directory unless you specifically require it. The root account has full access to every file on the system, regardless of the permissions attached to that file.

2.10.3 Data Verification

To create a checksum for a file, or to test a file against a checksum, use the *sha1sum* utility. SHA1 supersedes the older MD5 method, and you should always use SHA1. Linux also supply the GNU Privacy Guard (GnuPG) system for encrypting and digitally signing files, such as emails. Many documents refer to GnuPG as *gpg*, which is the name of the main GnuPG command. The files that you sign or encrypt with GnuPG are compatible with other applications that follow the *OpenPGP* standard. The Evolution email application automatically supports both signing and encrypting emails with GnuPG. Evolution is the default email application for several of the main Linux distributions, including Fedora, Novell Linux Desktop, Red Hat Enterprise Linux, and Ubuntu.

2.10.4 Encrypted Storage

Create one or more *encrypted volumes* for your sensitive files. Each volume is a single file which may enclose other files and directories of your choice. To access the

contents of the volume, you must provide the correct decryption password to a utility, which then makes the volume available as if it were a directory or drive. The contents of an encrypted volume cannot be read when the volume is not mounted. You may store or copy encrypted volume files as you wish without affecting their security. For example, the cross-platform *Truecrypt* utility enables you to create and access your encrypted volumes with all popular operating systems. In extreme cases, you may decide to encrypt an entire disk partition that holds or caches data, so that none of the contents may be read by unauthorized persons. On Linux you may use either *LUKS*, *CryptoFS* or *EncFS* to encrypt disks.

2.10.5 Remote Access

Majority Linux Distributions includes a version of *OpenSSH*, an implementation of the *SSH* standard for secure remote access. An SSH service uses strong encryption by default, and provides the following facilities:

- Remote command-line access
- Remote command execution
- Remote access to graphical software
- File transfers

2.10.6 OS Management

Majority of Linux distributions incorporate software management facilities based on *package* files and sets of specially prepared Web sites, known as *repositories* or *channels*. Package management utilities construct or update working copies of software from these packages, and execute any other setup tasks that packages require. Repositories enable the management tools to automatically provide the correct version of each software product, by downloading the required packages directly from the relevant repository. Most systems also use checksums and digital signature tests to ensure that packages are authentic and correct. In addition, package management tools can identify outdated versions of software by checking the software installed on a system against the packages in the repositories. This means that you can ensure that all of the supported software on your system does not suffer from known security vulnerability, simply by running the update routine.

2.10.7 Backup and System Recovery

You may easily restore program files for all of the software that is included with your distribution with the software management tools. In order to fully recover a system from accident, or deliberate compromise, you must also have access to copies of the data, configuration, and log files. These files require some form of separate backup mechanism.

All effective backup systems provide the ability to restore versions of your files from several earlier points in time. You may discover that the current system is damaged or compromised at any time, and need to revert to previous versions of key files, so keeping only one additional copy of a key file should not be considered an adequate backup. Majority of the Linux distributions provide a wide range of backup tools, and leave it to the administrator to configure a suitable backup arrangement for their systems.

2.10.8 Monitoring and Audit Facilities

On Linux systems, the *syslog* and *klogd* services record activity as it is reported by different parts of the system. The Linux kernel reports to klogd, whilst the other services and facilities on the system send log messages to a syslog service. Distributions provide several tools for reading and analyzing the system log files.

Several facilities on any UNIX-like system may also email reports and notifications directly to the root account, via the *SMTP* service. Edit the aliases file to redirect messages for root to another email address, and you will receive these emails at the specified address.

2.10.9 Application Isolation

Linux operating system provides several methods of limiting the ability of a program to affect either other running programs, or the host system itself.

- *Mandatory Access Control (MAC)* supplements the normal UNIX security facilities of a system by enforcing absolute limits that cannot be circumvented by any program or account.
- *Virtualization* enables you to assign a limited set of hardware resources to a virtual machine, which may be monitored and backed up by separate processes on the host system.
- *Linux Container* facilities, such as Docker, run processes within a generated filesystem and separate them from the normal processes of the host system
- The *chroot* utility runs programs within a specified working directory, and prevents them from accessing any other directory on that system.

2.10.10 Protection Against Virus and Malware

The security features of Linux or UNIX-like systems described above combine to form a strong defense against malware:

- Software is often supplied in the form of packages, rather than programs
- If you download a working program, it cannot run until you choose to mark the files as executable
- By default, applications such as the OpenOffice.org suite and the Evolution email client do not run programs embedded in emails or documents
- Web browsers require you to approve the installation of plug-ins
- Software vulnerabilities can be rapidly closed by vendors supplying updated packages to the repositories

2.11 WINDOWS VS LINUX

Microsoft Windows offered by Microsoft mainly targets the personal computing market. Windows OS has two versions i.e. 32 bits and 64 bits and is available in both clients as well as server versions. Windows was first released in the year 1985 and the latest client version of windows is Windows 10 which was released in the year 2015. Talking about the most recent server version, we have Windows server 2019.

Linux is a group of Unix-like operating systems based on the Linux kernel. It belongs to the family of free and open source software. It is usually packaged in a Linux distribution. Linux was first released in the year 1991. It is most commonly used for servers; however, a desktop version of Linux is also available which is very much popular and give a good competition to the Windows OS.

The comparison of both the popular Operating systems is compiled in the Table 2 as shown below:

Table 2: Windows Vs Linux Operating Systems

Features	Windows	Linux
Developer	Microsoft Corporation	Linus Torvalds, community.
Written in	C++, Assembly	Assembly language, C
OS family	Graphical Operating system family	Unix-like Operating System family
License	Proprietary commercial software	GPL(GNU General Public License) v2 and others
User Interface	Windows shell	Unix shell
Kernel type	Windows NT family has a hybrid kernel (combination of microkernel and monolithic kernel); Windows CE(Embedded compact) also have hybrid kernel; Windows 9x and earlier series has a monolithic kernel (MS-DOS).	Monolithic kernel (whole operating system works in the kernel space).
Source model	Closed source software; source available (through shared source initiative).	Open source software
Initial release	November 20, 1985 Windows is older than Linux.	September 17, 1991
Available in	138 languages	Multi-lingual
Platforms	ARM, IA-32, Itanium, x86-64, DEC Alpha, MIPS, PowerPC.	Alpha, H8/300, Hexagon, Itanium, m68k, Microblaze, MIPS, PA-RISC, PowerPC, RISC-V, s390, SuperH, NDS32, Nios II, OpenRISC, SPARC, ARC Unicore32, x86, Xtensa, ARM, C6x.
Package manager	Windows Installer (.msi), Windows Store (.appx).	Packaged in a Linux distribution (distro).

Booting	Can only be done from disk. the prime disk.	Can be done from any
Default command line	Windows PowerShell	BASH
Ease of use	Windows has a rich GUI and can be easily used by technical as well as non-technical persons. It is very simple and user-friendly.	CUI / GUI
Reliability	Windows is less reliable than Linux. Over the process recent years, Windows security, and reliability has been improved a lot. However, it still has some system instabilities and security weaknesses because of its oversimplified design.	Highly reliable and secure. It has a deep-rooted emphasis on management, system uptime.
Update	Windows update happens Users have full control when made. Installation takes less time and no reboot is required. Takes more time to install and requires a reboot.	an update in the current moment is inconvenient to users.
Access	Every user does not have access to the source code. Only the selected members of the group have access to the source code.	Users have access over the source code of kernel and can modify it accordingly. This gives a benefit that bugs in OS will be fixed faster. However, the drawback is that the developers may take undue advantage of the loophole.

☞ **Check Your Progress 1**

1) Mention the important features of LINUX Operating System.

.....

.....

.....

.....

2) Describe the architecture of LINUX.

.....

.....

.....

.....

2.12 SUMMARY

In this unit, we have discussed issues broadly related to features of LINUX OS, Architecture and components of LINUX, process management, memory management and file system in LINUX operating system. In this unit, we also discussed several theoretical concepts of LINUX system in detail, often useful in your lab for practice.

2.13 SOLUTIONS/ANSWERS

1) Linux is provided with lot of features, which are listed below.

   i. Linux operating system can work on different types of hardware devices and Linux kernel supports the installation of any kind of hardware platform.

   ii. **Linux is an Open Source software i.e., Linux** Source code is freely available. Linux also allow us to upgrade the source code. Many collaborative groups are working to upgrade and develop their own versions.

   iii. **Multiuser:** Linux operating system is a multiuser system, which means, multiple users can access the system resources such as RAM, Memory or Application programs at the same time.

   iv. **Multiprogramming:** Linux operating system is a multiprogramming system, which means multiple applications can run at the same time.

   v. **Hierarchical File System:** Linux operating system affords a standard file structure in which system files or user files are arranged.

   vi. **Shell:** Linux operating system offers a special interpreter program that can be used to execute commands of the OS. It can be used to do several types of operations like call application programs, and so on.

   vii. **Security:** Linux operating system offers user security systems using authentication features like encryption of data or password protection or controlled access to particular files.

2) **Linux Architecture:** Linux Operating System is a four-layered architecture.

   Hardware: All physical components of the computer that provides services to the user like CPU, RAM, Hard Disk, Monitor, Printer, etc., are known as hardware components of the computers.

   Kernel: This is the main component of the Linux operating system. Only through the Kernel, we can directly communicate with the hardware components.

   Shell: The shell layer is in between application layer and the kernel. Shell will take the input from the user applications and sends to the kernel in the form of instructions and takes output from the kernel and forwards it as a result to the user application.

   Applications: These are the utility programs for the users that run on shell. These applications like text editor, web browsers, spread sheet applications, etc.

2.14 FURTHER READINGS

1. Richard Pertersen, Linux: The Complete Reference, Sixth Edition, McGraw Hill, 2017.
2. MachteltGarrels, Introduction to Linux: A Hands-On Guide, UNIX Academy Publicaions, 2007.
3. Jason Cannon, Linux for Beginners: An Introduction to the Linux Operating System and Command Line, Kindle Edition, 2013.
4. Linux System Programming, Robert Love, O'Reilly, 2014.
5. K. Srirengan, *Understanding UNIX*, PHI, 2002.

UNIT 3 CASE STUDY - ANDROID

Structure

3.0 INTRODUCTION

Across the globe, in more than 190 countries, hundreds of millions of mobile devices are powered by Android operating system. It has strong user base and conquered around 75% of the global market share by the end of 2020. Google sponsored the project at initial stages and in the year 2005, it acquired the whole company. In September 2008, the first Android-powered device was launched in the market. It's popular because it has long list of features, user-friendly, has huge community support, provides a greater extent of customization, and a large number of companies build Android-compatible smart phones. At first, the purpose of Android was thought of as a mobile operating system. However, with the advancement of code libraries and its popularity among developers of the divergent domain, Android becomes an absolute set of software for all devices like tablets, wearables, set-top boxes, smart TVs, notebooks, etc. Android is an open-source operating system based on modified version of Linux kernel with a Java programming interface for mobile devices such as Smartphone (Touch Screen Devices who supports Android OS) as well for Tablets too.

This unit provides a case study on Android Operating System. It provides basic structure, features, process, file and memory management techniques it follows.

3.1 OBJECTIVES

After completing this unit, you will be able to:

- Understand the various features of the Android system
- Know the evolution of the Android OS

- Describe the architecture of the ANDROID OS.
- Know the Process and Memory management approaches
- Understand the File management in Android
- Identify the security features in Android

3.2 SALIENT FEATURES OF ANDROID OS

Android is a powerful open-source operating system that provides enormous features and some of them are listed below:

- **User Interface:** Android provides a great user interface which allow user to communicate with his device. It has user friendly UI which allow the user ease of access and can learn and operate the device quickly.
- **Storage:** Uses "SQLite" to store the data. SQLite being a relational data base processing queries are fast.
- **File Manager:** Data storage related activities will be managed by the file manager.
- **Media:** Supports wide range of media like JPEF, PNG, GIF, BMP, Ogg, MIDI, MP3, ACC 5.1, Vorbis, WAV, AMR, AMR-WB etc.
- **Messaging:** Provides various messaging services like MMS, SMS.
- **Connectivity:** Android provide various range of connectivity like GSM.EDGE, CDMA, Wi-Fi, LTE, NFC WiMAX, UMTS, EV-DO, IDEN
- **GPS:** It contains multiple APIs to support location-tracking services such as GPS.
- **Multitasking:** Android provides multitasking capability for the user. They can operate multiple applications at the same time.
- **Multi touch:** Android support multi touch functionality where user is allowed to make multiple touches at the same time and the OS will process it this is majorly used for gaming.
- **Widgets:** Android OS provides widgets which allow the user to have the organize the apps based on their requirements. It also allows the user to add, remove, resize, expand and collapse based on users usage and their need.
- **Screen Capture:** Allows the user to capture the screen shot of their mobile screen.
- **Near Field Communication:** Android provides a short range of wireless connection, which allows various devices to communicate quickly within the range. It is majorly used in payment systems.
- **Recording Capability:** It supports multimedia hardware control to perform playback or recording using a camera and microphone.
- **Webkit Layout:** Android has an integrated open-source WebKit layout based web browser to support User Interface like HTML5, CSS3.

- **Virtual Reality:** It provides support for virtual reality or 2D/3D Graphics.
- **Customization:** Customization is possible in Android OS based on our requirements.

3.3 EVOLUTION OF ANDROID OS

Andy Rubin founded Android Inc in October 2003 and later Google acquired Android Inc. Its initial intention is for camera as they have less market, it later changed to support mobile devices. Google announced the development of Android operating system in 2007 and Android launches its first Android mobile device. Android Inc. was acquired by Google in August, 2005. Key employees of Android Inc., including Andy Rubin, Rich Miner and Chris White, stayed at the company after the acquisition. After acquisition, a team led by Rubin developed a mobile device platform implemented using Linux kernel. Google marketed the platform to handset makers and carriers on the promise of providing a flexible, upgradable system.

Android operating systems has several versions. Following are the list of these versions depicted in the table 1:

Table 1: Android Versions

Code Name	Version	Release Date
Apple Pie	Android 1.0	September 23, 2008
Banana Bread	Android 1.1	February 9, 2009
Cupcake	Android 1.5	April 30, 2009
Donut	Android 1.6	September 15, 2009
Eclair	Android 2.0 – 2.1	October 26, 2009
Froyo	Android 2.2 – 2.2.3	May 20, 2010
Gingerbread	Android 2.3 – 2.3.4	December 6, 2010
Honeycomb	Android 3.0.x – 3.2.x	February 22, 2011
Ice Cream Sandwich	Android 4.0 – 4.0.4	October 18, 2011
Jelly Bean	Android 4.1 – 4.1.2	July 9, 2012
Kitkat	Android 4.4 – 4.4.4	July 9, 2012
Lollipop	Android 5.0 – 5.1	October 17, 2014
Marshmallow	Android 6.0 – 6.0.1	October 5, 2015
Nougat	Android 7.0 – 7.1	August 22, 2016
Oreo	Android 8.0	August 21, 2017
Pie	Android 9.0	August 6, 2018
Android Q	Android 10.0	September 3, 2019
Android 11	Android 11.0	September 8, 2020

In the next section let us study the Layered Architecture of Android.

3.4 LAYERED ARCHITECTURE OF ANDROID OS

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram given at Figure 1.

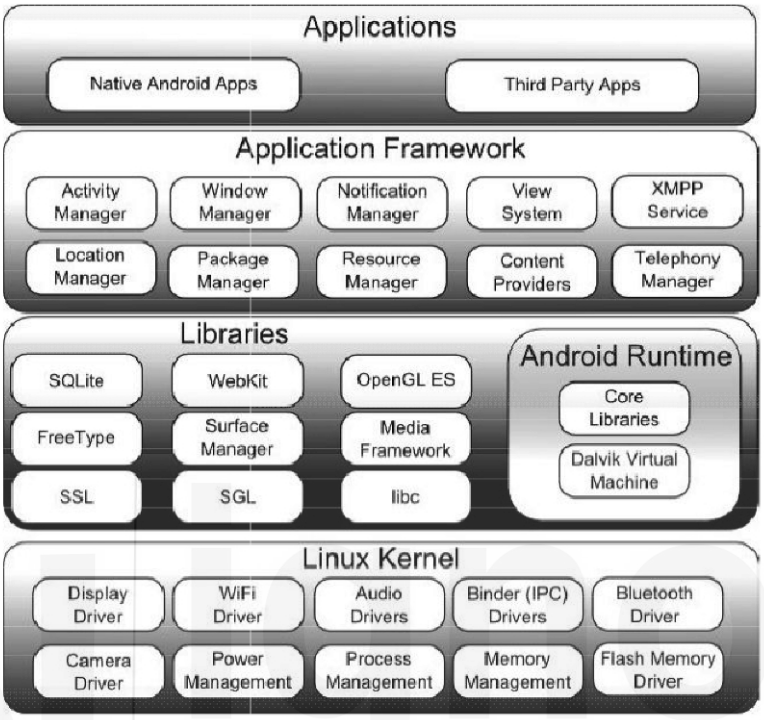

Figure 1: Architecture of Android OS (Source: developer.android.com)

3.4.1 Linux Kernel

Linux is the bottom layer. The bottom layer offers a level of abstraction between hardware and it contains all the essentials required for hardware drivers like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

3.4.2 Libraries

A rich set of libraries are present on top of Linux kernel. Along with open-source Library Web browser engine WebKit, a collection of popular libraries like libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc. Java based libraries are used for the purpose of development of Android OS. The application framework libraries in addition to those that facilitate user interface building, graphics drawing and database access.

A summary of some key core Android libraries available to the Android developer is as follows:

android.app	Provides access to the application model and is the cornerstone of all Android applications.
android.content	Facilitates content access, publishing and messaging between applications and application components.

android.database	Used to access data published by content providers and includes SQLite database management classes.
android.opengl	A Java interface to the OpenGL ES 3D graphics rendering API.
android.os	Provides applications with access to standard operating system services including messages, system services and inter-process communication
android.text	Used to render and manipulate text on a device display.
android.view	The fundamental building blocks of application user interfaces.
android.widget	A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
android.webkit	A set of classes intended to allow web-browsing capabilities to be built into applications.

3.4.3 Android Runtime

This section is the part of the Libraries layer which contains a key component called Dalvik Virtual Machine **(**Dalvik VM**)** which is an optimized Android library acts like a of Java Virtual Machine. The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables everyAndroid application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

3.4.4 Application Framework

Several high level services will be provided to the applications through Application Framework layer. These application services are available in the form of Java classes. Application developers can use these services in their applications.

The Android framework includes the following key services:

- **Activity Manager** " Controls all aspects of the application lifecycle and activity stack.
- **Content Providers** " Allows applications to publish and share data with other applications.
- **Resource Manager** " Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
- **Notifications Manager** " Allows applications to display alerts and notifications to the user.
- **Windows Manager** – Allows applications to apply various operations on windows like create, minimize, resize etc.
- **View System** " An extensible set of views used to create application user interfaces.

3.4.5 Applications

The top layer of Android Architecture is Applications-layer. Users have to install their own applications on this layer. Browser, Games, Contacts Books, etc., are some of the examples of such applications. The application layer runs within the Android run time using the classes and services made available from the application framework.

3.5 PROCESSES AND THREADS MANAGEMENT IN ANDROID

Android create process for each application, all the components are running from a main thread in low power and low memory. Android automatically manages the processes and kills the least used or inactive process.

3.5.1 Process Life Cycle

The Android system tries to maintain an application process for as long as possible, but eventually needs to remove old processes to reclaim memory for new or more important processes. To determine which processes to keep and which to kill, the system places each process into an "importance hierarchy" based on the components running in the process and the state of those components. Processes with the lowest importance are eliminated first, then those with the next lowest importance, and so on, as necessary to recover system resources. There are five levels in the importance hierarchy. The following list presents the different types of processes in order of importance:

(i) Foreground process

A process that is required for what the user is currently doing. A process is considered to be in the foreground if any of the following conditions are true:

- It hosts an Activity that the user is interacting with
- It hosts a Service that's bound to the activity that the user is interacting with.
- It hosts a Service that's running "in the foreground" and the service has called startForeground().
- It hosts a Service that's executing one of its lifecycle callbacks (onCreate(), onStart(), or onDestroy()).
- It hosts a BroadcastReceiver that's executing its onReceive() method.

Generally, only a few foreground processes exist at any given time. They are killed only as a last resort, if memory is so low and cannot continue to run. Generally, at that point, the device has reached a memory paging state, so killing some foreground processes is required to keep the user interface responsive.

(ii) Visible process

A process that doesn't have any foreground components, but still can affect what the user sees on screen. A process is considered to be visible if either of the following conditions are true:

- It hosts an Activity that is not in the foreground, but is still visible to the user (its onPause() method has been called).

- It hosts a Service that's bound to a visible (or foreground) activity.

A visible process is considered extremely important and will not be killed unless doing so is required to keep all foreground processes running.

(iii) Service process

A process that is running a service that has been started with the startService() method and does not fall into either of the two higher categories. Although service processes are not directly tied to anything the user sees, they are generally doing things that the user cares about such as playing music in the background or downloading data on the network, so the system keeps them running unless there's not enough memory to retain them along with all foreground and visible processes.

(iv) Background process

A process holding an activity that's not currently visible to the user (the activity's onStop() method has been called). These processes have no direct impact on the user experience, and the system can kill them at any time to reclaim memory for a foreground, visible, or service process. Usually there are many background processes running, so they are kept in an LRU (least recently used) list to ensure that the process with the activity that was most recently seen by the user is the last to be killed. If an activity implements its lifecycle methods correctly, and saves its current state, killing its process will not have a visible effect on the user experience, because when the user navigates back to the activity, the activity restores all of its visible state.

(v) Empty process

A process that doesn't hold any active application components is called empty process. The only reason to keep this kind of process alive is for caching purposes, to improve startup time the next time a component needs to run in it. The system often kills these processes in order to balance overall system resources between process caches and the underlying kernel caches.

Android ranks a process at the highest level it can, based upon the importance of the components currently active in the process.

3.5.2 Threads

When an application is launched, the system creates a thread of execution for the application, called "main". This thread is very important because it is in charge of dispatching events to the appropriate user interface widgets, including drawing events. It is also the thread in which your application interacts with components from the Android UI toolkit. This main thread is also sometimes called the UI thread.

The system does not create a separate thread for each instance of a component. All components that run in the same process are instantiated in the UI thread, and system calls to each component are dispatched from that thread. Consequently, methods that respond to system callbacks (such as onKeyDown() to report user actions or a lifecycle callback method) always run in the UI thread of the process.

For instance, when the user touches a button on the screen, your app's UI thread dispatches the touch event to the widget, which in turn sets its pressed state and posts an invalidate request to the event queue. The UI thread dequeues the request and notifies the widget that it should redraw itself.

When your app performs intensive work in response to user interaction, this single thread model can yield poor performance unless you implement your application properly. Specifically, if everything is happening in the UI thread, performing long operations such as network access or database queries will block the whole UI. When the thread is blocked, no events can be dispatched, including drawing events. From the user's perspective, the application appears to hang. Even worse, if the UI thread is blocked for more than a few seconds (about 5 seconds currently) the user is presented with the infamous "application not responding"" (ANR) dialog. The user might then decide to quit your application and uninstall it if they are unhappy. Additionally, the Android UI toolkit is not thread-safe.

Interprocess Communication

Android offers a mechanism for Interprocess Communication (IPC) using remote procedure calls (RPCs), in which a method is called by an activity or other application component, but executed remotely (in another process), with any result returned back to the caller. This entails decomposing a method call and its data to a level the operating system can understand, transmitting it from the local process and address space to the remote process and address space, then reassembling and reenacting the call there. Return values are then transmitted in the opposite direction. Android provides all the code to perform these IPC transactions, so you can focus on defining and implementing the RPC programming interface. To perform IPC, your application must bind to a service, using bindService().

In the next section we will study about the Memory Management in Android OS.

3.6 MEMORY MANAGEMENT IN ANDROID

The Android Runtime (ART) and Dalvik Virtual Machine(DVM) use paging and memory-mapping (*mmapping*) to manage memory. This means that any memory an app modifies; whether by allocating new objects or touching *mmapped* pages, they remains resident in RAM and cannot be paged out. The only way to release memory from an app is to release object references that the app holds, making the memory available to the garbage collector. That is with one exception: any files *mmapped* in without modification, such as code, can be paged out of RAM if the system wants to use that memory elsewhere.

3.6.1 Shared Memory

In order to fit everything it needs in RAM, Android tries to share RAM pages across processes. It can do so in the following ways:

- Each app process is forked from an existing process called Zygote. The Zygote process starts when the system boots and loads common framework code and resources (such as activity themes). To start a new app process, the system forks the Zygote process then loads and runs the app's code in the new process. This approach allows most of the RAM pages allocated for framework code and resources to be shared across all app processes.
- Most static data is *mmapped* into a process. This technique allows data to be shared between processes, and also allows it to be paged out when needed. Example static data include: Dalvik code (by placing it in a pre-

linked `.odex` file for direct *mmapping*), app resources (by designing the resource table to be a structure that can be *mmapped* and by aligning the zip entries of the APK), and traditional project elements like native code in `.so` files.

- In many places, Android shares the same dynamic RAM across processes using explicitly allocated shared memory regions. For example, window surfaces use shared memory between the app and screen compositor, and cursor buffers use shared memory between the content provider and client.

3.6.2 Types of Memory

Android devices contain three different types of memory: RAM, zRAM, and storage. Note that both the CPU and GPU access the same RAM.

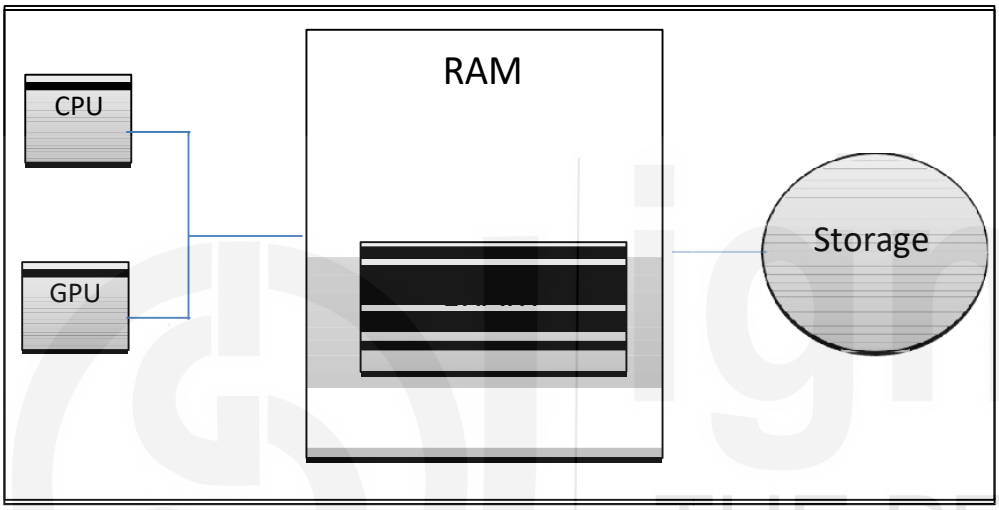

Figure 2: Types of Memory - RAM, zRAM, and Storage

- RAM is the fastest type of memory, but is usually limited in size. High-end devices typically have the largest amounts of RAM.
- zRAM is a partition of RAM used for swap space. Everything is compressed when placed into zRAM, and then decompressed when copied out of zRAM. This portion of RAM grows or shrinks in size as pages are moved into or taken out of zRAM. Device manufacturers can set the maximum size.
- Storage contains all of the persistent data such as the file system and the included object code for all apps, libraries, and the platform. Storage has much more capacity than the other two types of memory. On Android, storage isn't used for swap space like it is on other Linux implementations since frequent writing can cause wear on this memory, and shorten the life of the storage medium.

3.6.3 Memory Allocation Among Processes

The Android platform runs on the premise that free memory is wasted memory. It tries to use all of the available memory at all times. For example, the system keeps apps in memory after they've been closed so the user can quickly switch back to them. For this reason, Android devices often run with very little free memory. Memory management is vital to properly allocate memory among important system processes and many user applications.

3.6.4 Paging

RAM is broken up into *pages*. Typically each page is 4KB of memory. Pages are considered either *free* or *used*. Free pages are unused RAM. Used pages are RAM that the system is actively using, and are grouped into the following categories:

- ***Cached:*** Memory backed by a file on storage (for example, code or memory-mapped files). There are two types of cached memory:
- ***Private:*** Owned by one process and not shared.
- ***Clean:*** Unmodified copy of a file on storage, can be deleted by `kswapd` to increase free memory.
- ***Dirty:*** Modified copy of the file on storage; can be moved to, or compressed in, zRAM by `kswapd` to increase free memory or allows changes to be written back to the file in storage to increase free memory by `kswapd`, or explicitly using `msync()` or `munmap()or` Can be moved/compressed in zRAM by `kswapd` to increase free memory
- ***Shared:*** Used by multiple processes.
- ***Anonymous:*** Memory **not** backed by a file on storage (for example, allocated by `mmap()` with the `MAP_ANONYMOUS` flag set).

3.6.5 Low Memory Management

Android has two main mechanisms to deal with low memory situations:

- The kernel swap daemon
- Low-memory killer

(i) kernel swap daemon

The kernel swap daemon (kswapd) is part of the Linux kernel, and converts used memory into free memory. The daemon becomes active when free memory on the device runs low. The Linux kernel maintains low and high free memory thresholds. When free memory falls below the low threshold, kswapd starts to reclaim memory. Once the free memory reaches the high threshold, kswapd stops reclaiming memory. kswapd can reclaim clean pages by deleting them because they're backed by storage and have not been modified. If a process tries to address a clean page that has been deleted, the system copies the page from storage to RAM. This operation is known as *demand paging*.

kswapd can move cached private dirty pages and anonymous dirty pages to zRAM, where they are compressed. Doing so frees up available memory in RAM (free pages).

If a process tries to touch a dirty page in zRAM, the page is uncompressed and moved back into RAM. If the process associated with a compressed page is killed, then the page is deleted from zRAM. If the amount of free memory falls below a certain threshold, the system starts killing processes.

(ii) Low-Memory Killer

Many times, kswapd cannot free enough memory for the system. In this case, the system uses onTrimMemory() to notify an app that memory is running low and that it should reduce its allocations. If this is not sufficient, the kernel starts killing processes to free up memory. It uses the low-memory killer (LMK) to do this.

To decide which process to kill, LMK uses an "out of memory" score called oom_adj_score to prioritize the running processes. Processes with a high score are killed first. Background apps are first to be killed, and system processes are last to be killed. The following table lists the LMK scoring categories from high-to-low. Items in the highest-scoring category, in row one, will be killed first:

These are descriptions for the various categories in the table above:

- **Background Apps:** Apps that were run previously and are not currently active. LMK will first kill background apps starting with the one with the highest oom_adj_score.
- **Previous App:** The most recently-used background app. The previous app has higher priority (a lower score) than the background apps because it is more likely the user will switch to it than one of the background apps.
- **Home App:** This is the launcher app. Killing this will make the wallpaper disappear.
- **Services:** Services are started by applications and may include syncing or uploading to the cloud.
- **Perceptible Apps:** Non-foreground apps that are perceptible to the user in some way, such as running a search process that displays a small UI or listening to music.
- **Foreground App**: The app currently being used. Killing the foreground app looks like an application crash which might indicate to the user that something is going wrong with the device.
- **Persistent (services):** These are core services for the device, such as telephony and wifi.
- **System:** System processes. As these processes are killed, the phone may appear to reboot.
- **Native:** Very low-level processes used by the system (for example, kswapd).

3.6.6 Garbage Collection

A managed memory environment, like the ART or DVM, keeps track of each memory allocation. Once it determines that a piece of memory is no longer being used by the program, it frees it back to the heap, without any intervention from the programmer. The mechanism for reclaiming unused memory within a managed memory environment

is known as *garbage collection*. Garbage collection has two goals: find data objects in a program that cannot be accessed in the future; and reclaim the resources used by those objects.

Android's memory heap is a generational one, meaning that there are different buckets of allocations that it tracks, based on the expected life and size of an object being allocated. For example, recently allocated objects belong in the *Young generation*. When an object stays active long enough, it can be promoted to an older generation, followed by a permanent generation.

3.7 FILE MANAGEMENT IN ANDROID

Android uses different partitions to handle files and folders on the system like windows operating system. Each of these partitions has its own functionality. Mainly, there are six specific partitions in the file system of Android devices. Each model may have their own organization for this file system partitioning. But most of the Android Devices have the following list of partitions logically in common like: */boot, /system, /recovery, /data, /cache* and */misc*. The following are two separate partitions for the SD card of Android Devices */sdcard* and */sd-ext.*

/boot

- This is the boot partition of your Android device, as the name suggests.
- It includes the android kerneland the ramdisk.
- The device will not boot without this partition.
- Wiping this partition from recovery should only be done if absolutely required and once done, the device must NOT be rebooted before installing a new one, which can be done by installing a ROM that includes a /boot partition.

/system

- As the name suggests, this partition contains the entire Android OS.
- This includes the Android GUI and all the system applications that come pre-installed on the device.
- Wiping this partition will remove Android from the device without rendering it unbootable, and we can still be able to place the phone into recovery or bootloader mode to install a new ROM.

/recovery

- This is specially designed for backup.
- The recovery partition can be considered as an alternative boot partition, that lets the device boot into a recovery console for performing advanced recovery and maintenance operations on it.

/data

- It is called user data partition.
- This partition contains the user's data like your contacts, sms, settings and all android applications that you have installed.

- While we are doing factory reset on our device, this partition will wipe out, then the device will be in the state, when we use for the first time, or the way it was after the last official or custom ROM installation.

/cache

- This is the partition where Android stores frequently accessed data and app components.
- Wiping the cache doesn't affect our personal data but simply gets rid of the existing data there, which gets automatically rebuilt as we continue using the device.

/misc

- This partition contains miscellaneous system settings in form of on/off switches.
- These settings may include CID (Carrier or Region ID), USB configuration and certain hardware settings etc.
- This is an important partition and if it is corrupt or missing, several of the device's features will not function normally.

/sdcard

- This is not a partition on the internal memory of the device but rather the SD card.
- In terms of usage, this is our storage space to use as you see fit, to store media, documents, ROMs etc. on it.
- Wiping it is perfectly safe as long as we backup all the data you require from it, to the computer first.
- Though several user-installed apps save their data and settings on the SD card and wiping this partition will make we lose all that data.

/sd-ext

- This is not a standard Android partition, but has become popular in the custom ROM scene.
- It is basically an additional partition on your SD card that acts as the /data partition.
- It is especially useful on devices with little internal memory allotted to the / data partition.
- Thus, users who want to install more programs than the internal memory allows can make this partition and use it for installing their apps.

3.8 SECURITY FEATURES IN ANDROID

Android incorporates industry-leading security features and works with developers and device implementers to keep the Android platform and ecosystem safe. A robust security model is essential to enable a vigorous ecosystem of apps and devices built on and around the Android platform and supported by cloud services. As a result, through

its entire development lifecycle, Android has been subject to a rigorous security program. The key components of the Android Security include:

3.8.1 System and kernel security

At the operating system level, the Android platform provides the security of the Linux kernel, as well as a secure inter-process communication (IPC) facility to enable secure communication between applications running in different processes. These security features at the OS level ensure that even native code is constrained by the Application Sandbox. Whether that code is the result of included application behavior or an exploitation of application vulnerability, the system is designed to prevent the rogue application from harming other applications, the Android system, or the device itself.

The foundation of the Android platform is the Linux kernel. Linux has become a stable and secure kernel trusted by many corporations and security professionals. As the base for a mobile computing environment, the Linux kernel provides Android with several key security features, including:

- A user-based permissions model
- Process isolation
- Extensible mechanism for secure IPC
- The ability to remove unnecessary and potentially insecure parts of the kernel

As a multiuser operating system, a fundamental security objective of the Linux kernel is to isolate user resources from one another. The Linux security philosophy is to protect user resources from one another.

3.8.2 Verified boot

Android 6.0 and later supports verified boot and device-mapper-verity. Verified boot guarantees the integrity of the device software starting from a hardware root of trust up to the system partition. During boot, each stage cryptographically verifies the integrity and authenticity of the next stage before executing it. Android 7.0 and later supports strictly enforced verified boot, which means compromised devices cannot boot.

3.8.3 File System Permissions

In a UNIX-style environment, file-system permissions ensure that one user cannot alter or read another user's files. In the case of Android, each application runs as its own user. Unless the developer explicitly shares files with other applications, files created by one application cannot be read or altered by another application.

3.8.4 Platform Security

Android seeks to be the most secure and usable operating system for mobile platforms by repurposing traditional operating system security controls to Protect app and user data, Protect system resources (including the network),Provide app isolation from the system, other apps, and from the user. To achieve these objectives, Android provides these key security features:

- Robust security at the OS level through the Linux kernel
- Mandatory app sandbox for all apps

- Secure interprocess communication
- App signing
- App-defined and user-granted permissions

3.8.5 Program Level Security Features

- **Design review:** The Android security process begins early in the development lifecycle with the creation of a rich and configurable security model and design. Each major feature of the platform is reviewed by engineering and security resources, with appropriate security controls integrated into the architecture of the system.
- **Penetration testing and code review:** During the development of the platform, Android-created and open source components are subject to vigorous security reviews. These reviews are performed by the Android Security Team, Google's Information Security Engineering team, and independent security consultants. The goal of these reviews is to identify weaknesses and possible vulnerabilities well before major releases, and to simulate the types of analysis that are performed by external security experts upon release.
- **Open source and community review:** AOSP enables broad security review by any interested party. Android also uses open source technologies that have undergone significant external security review, such as the Linux kernel. Google Play provides a forum for users and companies to provide information about specific apps directly to users.
- **Incident response:** Even with these precautions, security issues may occur after shipping, which is why the Android project has created a comprehensive security response process. Full-time Android security team members monitor the Android-specific and the general security community for discussion of potential vulnerabilities and review security bugs filed on the Android bug database. Upon the discovery of legitimate issues, the Android team has a response process that enables the rapid mitigation of vulnerabilities to ensure that potential risk to all Android users is minimized. These cloud-supported responses can include updating the Android platform (AOSP updates), removing apps from Google Play, and removing apps from devices in the field.
- **Monthly security updates:** The Android security team provides monthly updates to Google Android devices and all our device manufacturing partners.

3.8.6 Cryptography

Android provides a set of cryptographic APIs for use by applications. These include implementations of standard and commonly used cryptographic primitives such as AES, RSA, DSA, and SHA. Additionally, APIs are provided for higher level protocols such as SSL and HTTPS.

3.8.7 User Security Features

This can be categorized into File System Encryption and Password Protection.

(i) File System Encryption

Android 3.0 and later provides full file-system encryption, so all user data can be

encrypted in the kernel. Android 5.0 and later supports full disk encryption. Full-disk encryption uses a single key—protected with the user's device password—to protect the whole of a device's user data partition. Upon boot, users must provide their credentials before any part of the disk is accessible.

Android 7.0 and later supports file-based encryption. File-based encryption allows different files to be encrypted with different keys that can be unlocked independently.

(ii) Password Protection

Android can be configured to verify a user-supplied password prior to providing access to a device. In addition to preventing unauthorized use of the device, this password protects the cryptographic key for full file system encryption. Use of a password and/or password complexity rules can be required by a device administrator.

3.8.8 Device Security

Android 2.2 and later provide the Android Device Administration API, which provides device administration features at the system level. For example, the built-in Android Email application uses the APIs to improve Exchange support. Through the Email application, Exchange administrators can enforce password policies including alphanumeric passwords or numeric PINs across devices. Administrators can also remotely wipe (that is, restore factory defaults on) lost or stolen handsets.

☞ **Check Your Progress 1**

1) Mention and explain briefly the main building blocks of Android platform.

.....

.....

.....

.....

2) Describe the new IPC mechanisms provided by Android.

.....

.....

.....

.....

3.9 SUMMARY

Android is a powerful open-source operating system which provides a lot of great features. Its open-source and we can customize the OS based on our requirements. It supports connectivity for GSM, CDMA, WIFI, NFC, Bluetooth, etc. for telephony or data transfer. It will allow us to make or receive a calls / SMS messages and we can send or retrieve data across mobile networks. By using WIFI technology we can pair with other devices using apps. Android has multiple APIs to support location-based services such as GPS. We can perform all data storage related activities by using lightweight database SQLite. It has a wide range of media supports like AVI, MKV,

FLV, MPEG4, etc. to play or record a variety of audio/video and having a different image format like JPEG, PNG, GIF, BMP, MP3, etc. . It has extensive support for multimedia hardware control to perform playback or recording using camera and microphone. It has an integrated open-source WebKit layout based web browser to support HTML5, CSS3. It supports a multi-tasking, we can move from one task window to another and multiple applications can run simultaneously. It will give a chance to reuse the application components and the replacement of native applications. We can access the hardware components like Camera, GPS, and Accelerometer. It has support for 2D/3D Graphics.

In this unit, we have discussed features of Android OS, Architecture, process management, memory management, file system and security aspects in Android operating system.

3.10 SOLUTIONS / ANSWERS

Check Your Progress 1:

1) The main Android platform building blocks are:

- **Device hardware:** Android runs on a wide range of hardware configurations including mobile phones, tablets, watches, automobiles, smart TVs, OTT gaming boxes, and set-top-boxes. Android is processor-agnostic, but it takes advantage of some hardware-specific security capabilities such as ARM eXecute-Never.
- **Android operating system:** The core operating system is built on top of the Linux kernel. All device resources, like camera functions, GPS data, Bluetooth functions, telephony functions, and network connections are accessed through the operating system.
- **Android Application Runtime:** Android apps are most often written in the Java programming language and run in the Android runtime (ART). However, many apps, including core Android services and apps, are native apps or include native libraries. Both ART and native apps run within the same security environment, contained within the Application Sandbox. Apps get a dedicated part of the file system in which they can write private data, including databases and raw files.

  Android apps extend the core Android operating system. There are two primary sources for apps:

- **Preinstalled apps:** Android includes a set of preinstalled apps including phone, email, calendar, web browser, and contacts. These function as user apps and they provide key device capabilities that can be accessed by other apps. Preinstalled apps may be part of the open source Android platform, or they may be developed by a device manufacturer for a specific device.
- **User-installed apps:** Android provides an open development environment that supports any third-party app. Google Play offers users hundreds of thousands of apps.

  Processes can communicate using any of the traditional UNIX-type mechanisms. Examples include the file-system, local sockets, or signals. However, the Linux permissions still apply.

2) Android also provides new IPC mechanisms:

- **Binder**: A lightweight capability-based remote procedure call mechanism designed for high performance when performing in-process and cross-process calls. Binder is implemented using a custom Linux driver.
- **Services**: Services (discussed above) can provide interfaces directly accessible using binder.
- **Intents**: An Intent is a simple message object that represents an "intention" to do something. For example, if your application wants to display a web page, it expresses its "Intent" to view the URL by creating an Intent instance and handing it off to the system. The system locates some other piece of code (in this case, the Browser) that knows how to handle that Intent, and runs it. Intents can also be used to broadcast interesting events (such as a notification) system-wide.
- **ContentProviders**: A ContentProvider is a data storehouse that provides access to data on the device; the classic example is the ContentProvider that is used to access the user's list of contacts. An application can access data that other applications have exposed via a ContentProvider, and an application can also define its own ContentProviders to expose data of its own.

While it is possible to implement IPC using other mechanisms such as network sockets or world-writable files, these are the recommended Android IPC frameworks. Android developers will be encouraged to use best practices around securing users' data and avoiding the introduction of security vulnerabilities.

3.11 FURTHER READINGS

1. Barry Burd, Android Application Development All-in-One for Dummies, Second Edition, Wiley, 2015.
2. Pradeep Kothari, Android Application Development, Black Book, Kindle Edition(Dreamtech Press), 2019.
3. Pratiyush Guleria,Android for Beginners: Learn Step-by-Step, BPB Publications, 2018.
4. developer.android.com

UNIT 4 CASE STUDY - iOS

Structure

4.0 INTRODUCTION

iPhone operating system (iOS) is a mobile operating system developed by Apple Inc. used for apple handheld devices. It is used in devices like- iPhone, iPad and iPod. It is the second most widely used mobile operating system. It supports features like direct manipulation and can respond to various types of user gestures. It is proprietary and closed source and derived from macOS. Software development kit (SDK) is provided to create applications. It includes interfaces for developing, running and testing applications. Apps can be written using system frameworks and Objective-C programming language.

Initial three versions were introduced with the name iPhone OS. From fourth version, they renamed it to iOS. With each new version, new features and apps were added. iOS 13 is latest version which was released in 2019. Its major feature is dark mode and new Map application with street view capability. It also enhanced its previous apps and features like-Siri, Heath map and others. iOS 14 is about to release in 2020.

4.1 OBJECTIVES

After going through this unit, you should be able to:

- Understand the basic functions of iOS Operating System
- Know the history of iOS operating system
- Understand the process management in iOS and can compare with other OS
- Understand the memory management approaches in iOS
- Understand the File management in iOS
- Understand the security features in iOS

4.2 FEATURES OF IOS

iOS is the operating system for iPhone, iPad and other Apple mobile devices. Based on Mac OS, the operating system which runs Apple's line of Mac desktop and laptop computers, Apple iOS is designed for easy, seamless networking between Apple products. iOS support extremely good features and some of the common features are:

- Multitasking – it allows multiple tasks to be executed concurrently. This feature allows various applications to run in background like notifications, VoIP, audio, Bluetooth access, app updates and may more.
- SpringBoard – It is used for managing home screen.
- Gesture recognition
- Wifi, Bluetooth and support for VPN
- Access to Apple App store
- Support for integrated search – it allows files to be search simultaneously
- Safari browser
- Front and read camera with videos capturing facility
- Siri – it is an intelligent personal assistant feature which can take voice queries and can give voice responses and recommendations, used for setting reminders etc.
- Game center – it is multiplayer gaming network available online.
- Compatible with iCloud – iCloud is cloud service provided by Apple.
- Push email service – Apples's email server allows mails to be delivered as they arrive.
- Accelerometer, gyroscope, and magnetometer – these are sensor interfaces used to listen various events.
- Apple pay – it is payment technology which can store credit card details to pay for services.

- Services like Maps, contacts, web pages, messages, location
- Various security features – face ID, pass code, 2 factor authentication.
- HomePod – it can identify family members by voice and can handoff calls , music etc on other devices.
- HomeKit – it is home automation controlling system.
- CarPlay – this allows interacting with iOS during drive. Also allows access tom phone apps.

Apple keeps making iOS 14 and iPadOS 14 even better by adding valuable new capabilities and features. Most recently, Apple released iOS 14.4, adding a new workout to Fitness Plus for Apple Watch owners. It also added a new Unity watch face in February to celebrate Black History Month. The update also included a series of security fixes for vulnerabilities that were actively being exploited.

iOS 14.4 follows the addition of Apple ProRaw photos to the iPhone 12 Pro and 12 Pro Max. Those new features join an already impressive list of capabilities that Apple brought to its mobile devices with the release of iOS 14 in September. iOS 14.5 is currently available in beta and is shaping up to be a significant update for iPhone owners.

4.3 EVOLUTION OF IOS

iOS was first introduced with iPhone in Jan 2007 and released in June 2007. At first introduction Steve Jobs claimed it to be running OS X and running desktop class application but during release introduced it with the name "iPhone OS". Initially, there was no support for third party applications. In March 2008, Apple announced software development kit for iPhone. In July 2008, iOS app store opened with 500 applications which increased to 3000 in Sept 2008 and after successive growth through the years increased to 2.2 million in 2017. It is also estimated to reach 5 million by 2020. iOS has seen a lot of changes since its inception. Following are the important milestones in iOS:

- iOS was first introduced with iPhone in jan 2007 and released in june 2007.
- Apple's iOS first SDK was released on March 6, 2008.
- The initial release was named iPhone OS which was later changed to iOS on June 7, 2010.
- iPhone OS 1 was released on March 6, 2008, and is the first version of the popular operating system. The support for iPhone OS 1 ended after two years, i.e., 2010.
- iPhone OS 2, as the name suggest, is the 2nd big release for the iOS. The release was done in conjunction with iPhone 3G, and anyone with the previous version can easily upgrade to the latest version. Also, this version introduced the App store, which becomes the hub for installing new apps. New SDK was also released for developers with support ending in 2011.
- The third big release was Apple iOS 3. It came into existence in June 2009 with support ending in late 2012. New features such as copy, paste, etc. are added to the OS.

The next version is iOS 4 and is released on June 21, 2010. Clearly, this is one of the big releases for iOS as it dropped old device support instead of supporting the latest devices with multitasking features.

- iOS 5 was released on June 6, 2001. It brought support for iPad Touch (3rd generation) and iPad (1st generation).
- iOS 6 went live on September 19, 2012, for the 4th generation Apple devices.
- iOS 7 was released for public on September 18, 2013. It supported two new phones by Apple, the Apple iPhone 5S and iPhone 5C.
- Just like the old release, iOS 8 released for public on September 9, 2014, with support for their best phone devices, the iPhone 6 and iPhone 6 Plus. They dropped support for older devices.
- iOS 9 was made public on September 16, 2015. Apple changed how they support legacy hardware and iOS 9 became the first Apple OS that supported 22 devices.
- iOS 10 was announced on June 13, 2016 at WWDC(Worldwide Developers Conference event and was released to public in September, 2016 along with iPhone 7 and iPhone 7 plus.
- iOS 11 was made public on September, 2017 along with iPhoen 8 and iPhone 8 Plus. It has dropped 32-bit applications making iOS 11 as a 64-bit OS that only runs 64-bit apps.
- iOS 12 was made public on September 2018 along with iPhone XS, iPhone XS Max, iPhone XR.
- Apple announced iOS 13 and made public on September, 2019. The principal features include Dark Mode and Memoji support. The NFC(Near Field Communication) framework supports reading several types of contactless smartcards and tags.
- Apple released iOS 14 and iPadOS 14 on 9th July 2020. All devices that support iOS 13 also support iOS 14. Some new features includes widgets that be now be placed directly on the home-screen, along with the App library which automatically categorizes apps into one page, Picture in Picture, Car- key technology to unlock and start a car with NFC. It also allow the user to have incoming calls shown in banners rather than taking up the whole screen. As on date (March, 2021) 14.1 is available in beta 3 version.

4.4 ARCHITECTURE OF IOS

iOS architecture is written in Objective-C language and comprised of four layers(layered architecture). It consists of a stack of four layers – Core OS, Core services, media layer, and Cocoa Touch as shown in Figure 1. Apps installed on system communicate with the iOS which in turn communicates with the hardware. Bottom level layers in the iOS stack are responsible for basic services while the upper layers are responsible for providing interface and graphics designing.

Apple provides system interfaces called **Frameworks** which is a package that stores dynamic shared libraries. It contains various resources like – header files, supported

apps, images. Each layer of iOS contains different set of frameworks that can be used to developers to design apps.

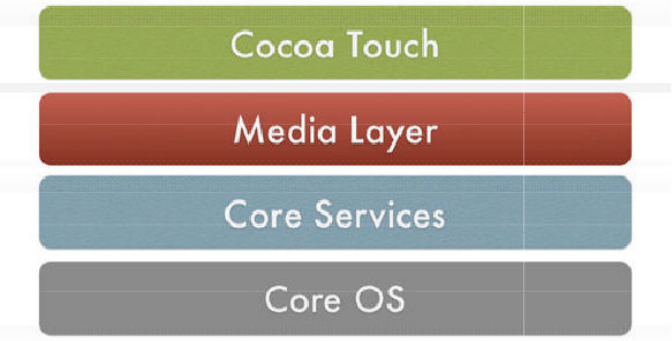

Fig 1. iOS Layered Architecture

Let us learn more about the layered architecture shown in the above figure.

4.4.1 Core OS

It is the lowest layer in the iOS architecture hence responsible for providing basic low level features. The frameworks present in this layer are:

- Security services framework
- Authentication framework
- Bluetooth framework
- External accessory framework
- Accelerate framework

4.4.2 Core Services

It is the layer at the top of core Os layer. The various frameworks available at this layer are:

- Core data Framework - It is used for managing Model View Controller (MVC) app.
- Core Foundation framework - It gives interfaces used to manage data, services and features.
- Address book framework - Used for accessing user's contacts database.
- Cloud Kit framework -Allows movement of data between apps and iCloud.
- Core Location framework - Used to give location and heading information to apps.
- Core Motion Framework - Used to access motion based data and accelerometer based information.
- Healthkit framework - Used to handle health oriented information of user
- Homekit framework - Used to connect and control user's home devices.
- Social framework - Used to handle social media accounts of user.
- StoreKit framework - Used to provides In -App Purchase.

4.4.3 Media Layer

This is the third layer from the bottom. It is used for providing audio, video and graphics features.

Audio Frameworks

- Media Player Framework - It is used to handle user's playlist.
- AV Foundation - It is used for recording and playback of audio and video.
- OpenAL- It is standard technology for providing audio. Video Frameworks
- AV Kit - This interface is used for video presentation.
- AV Foundation - It gives advanced video recording and playback capability.
- Core Media - It is used to describe low level data types and interfaces for operating media.

Graphics Frameworks

- Core Graphics framework - It is used for custom 2D vector and image based rendering.
- Core Animation - It is used for adding animation in apps.
- Core Images - It is used for gives for controlling video and images
- UIKit Graphics - It is used for designing images and animating content of views.
- OpenGl ES and GLKit - It is used for managing2D and 3D rendering.

4.4.4 Cocoa touch layer

It is the top most layer in iOS architecture. It is used for providing interactive interfaces to user. The various frameworks present in this layer are:

- UIKit Framework – It is used for designing graphical, event-driven apps.
- EventKit framework - It is used for providing system interfaces for viewing and altering calendar related events
- GameKit Framework - It allows users to share game related information online.
- MapKit Framework - It allows maps to be included in user interface.
- PushKitFramework- It provides support for VoIP apps.
- Twitter Framework - It allows access to Twitter service.

4.5 iOS KERNEL ARCHITECTURE

iOS kernel is XNU kernel of Darwin. Initially it was used only for MacOS and later introduced as free and open source part of Darwin operating system. It is a hybrid kernel which supports both monolithic and microkernel.

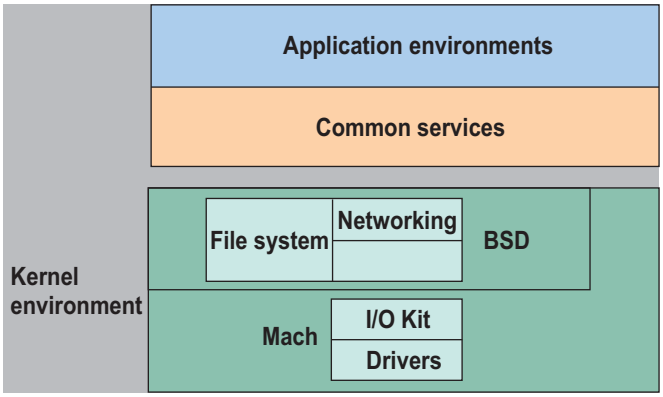

Fig 2: XNU kernel Architecture
(Source: developer.apple.com)

The foundation layer of Darwin and OS X is composed of several architectural components, as shown in Figure 2. These components form the *kernel environment*. In OS X, however, the kernel environment contains much more than the Mach kernel itself. The OS X kernel environment includes the Mach kernel, BSD, the I/O Kit, file systems, and networking components. These are often referred to collectively as the kernel. Each of these components is described briefly in the following sections. For further details, refer to the specific component chapters or to the reference material listed in the bibliography.

Because OS X contains three basic components (Mach, BSD, and the I/O Kit), there are also frequently as many as three APIs for certain key operations. In general, the API chosen should match the part of the kernel where it is being used, which in turn is dictated by what your code is attempting to do. The remainder of this chapter describes Mach, BSD, and the I/O Kit and outlines the functionality that is provided by those components.

Mach

Mach manages processor resources such as CPU usage and memory, handles scheduling, provides memory protection, and provides a messaging-centered infrastructure to the rest of the operating-system layers. The Mach component provides untyped interprocess communication (IPC), remote procedure calls (RPC), scheduler support for symmetric multiprocessing (SMP), support for real-time services, virtual memory support, support for pagers and modular architecture.

BSD

Above the Mach layer, the BSD layer provides "OS personality" APIs and services. The BSD layer is based on the BSD kernel, primarily *FreeBSD*. The BSD component provides file systems, networking (except for the hardware device level), UNIX security model, syscall support, the BSD process model, including process IDs and signals, FreeBSD kernel APIs, many of the *POSIX* APIs, kernel support for *pthreads* (POSIX threads).

Networking

OS X networking takes advantage of BSD's advanced networking capabilities to provide support for modern features, such as Network Address Translation (*NAT*) and *firewalls*. The networking component provides 4.4BSD TCP/IP stack and socket APIs, support for both IP and DDP (AppleTalk transport), multihoming, routing,

multicast support, server tuning, packet filtering, Mac OS Classic support (through filters).

File Systems

OS X provides support for numerous types of file systems, including *HFS*, *HFS+*, *UFS*, *NFS*, *ISO 9660*, and others. The default file-system type is HFS+; OS X boots (and "roots") from HFS+, UFS, ISO, NFS, and UDF. Advanced features of OS X file systems include an enhanced Virtual File System (*VFS*) design. VFS provides for a layered architecture (file systems are *stackable*). The file system component provides *UTF-8* (Unicode) support and increased performance over previous versions of Mac OS.

I/O Toolkit

The I/O Kit component provides true plug and play, dynamic device management, dynamic (on-demand) loading of drivers, power management for desktop systems as well as portables and multiprocessor capabilities.

4.6 PROCESSES AND THREADS MANAGEMENT

Every process in an application is created using one or more threads. A thread represents single path of execution. Execution of a process starts with a single thread and can later spawn multiple threads, where each thread performs a specific task.

All threads of a single process have same access rights and also shares same address space (virtual memory). They can also communicate with each other and other processes. Every thread has its own stack of execution and scheduled for execution by kernel separately.IOS has the capability to execute multiple programs parallel but most of the execution is done in background and does not require continuous processing. The application that runs in foreground keeps the processor and other resources busy.

4.6.1 Threading Packages

Applications may require creation of threads. IOS supports thread management and synchronization along with other new technologies. The technologies used to manage thread in iOS can be POSIX threads or Cocoa threads.

POSIX threads are based on C language. This must be used when application needs to be designed for multiple platforms. Here threads are managed using *pthread* library. Communication between threads can be done using ports, shared memory or conditions. For Cocoa applications, threads are created using NSThread class. It allows detached thread to be created. Detached thread is the one in which system automatically reclaims thread's resources when it terminates.

4.6.2 Threads Alternatives

Threads are low level doing concurrency and managing synchronization becomes difficult for application development. Also the use of thread adds processor and memory overheads. iOS hence offers alternatives to thread programming. Execution of concurrent tasks or processes on iOS is managed by asynchronous designing approach like traditional thread approach. Applications should only define specific tasks instead of creating threads as threads are low level and should leave it to system. Threads when

managed by system, gives scalability to applications. The various techniques provided by iOS to manage tasks asynchronously are:

- **Grand Central Dispatch (GCD)**

  In this approach user only defines the task to be done and add it to the appropriate dispatch queue. GCD then creates threads as needed and schedules them. Since in this approach thread management and execution is done by system, it is more efficient than traditional threads approach.

- **Operation Queues**

  They are similar to dispatch queue and are object-C objects. User defines the task to execute and add it to operation queue. Scheduling and execution is then done by operation queue.

Dispatch queues: Dispatch queue are used for execution of customized tasks. It executes the task serially or concurrently in first-in first-out order. Serial dispatcher waits for the task to complete before de-queuing, whereas concurrent dispatcher does not wait for the tasks to finish before starting new tasks.

Operation Queues: In operation queues the order of execution of task is dependent upon many factors. One such factor is the dependence of one task on the completion of another task. User when defining a task should also configure dependency.

The various advantages of using dispatch and operation queues in place of threads offer various advantages:

1. No need to store thread stack hence reduces memory requirements
2. User need not to configure and manage threads.
3. User need not to schedule threads.
4. It is easy to write code as compared to threads.

4.7 MEMORY MANAGEMENT

For effective and efficient utilization of memory, memory management is required by the iOS. Resource needs to be free up when they are not required. In iOS memory used by applications is managed by maintaining the life cycle of objects and freeing them when not required. To identify that an object is no more required; its object graph is created. The group of objects that are related to each other forms **object graph**. In this, objects can be related by the direct reference or indirect reference.

4.7.1 Application Memory Management

Applications memory management can be done by Object - C by two methods:

i. Manual Retain Release – In this method user can themselves manage memory by keeping track of objects created by them. This is implemented using **Reference Count Model.** (for versions till iOS-5)

ii. Automatic Reference Counting – In this feature, provided by compiler, system uses Reference counting model by calling methods for user at compile time. (for new versions)

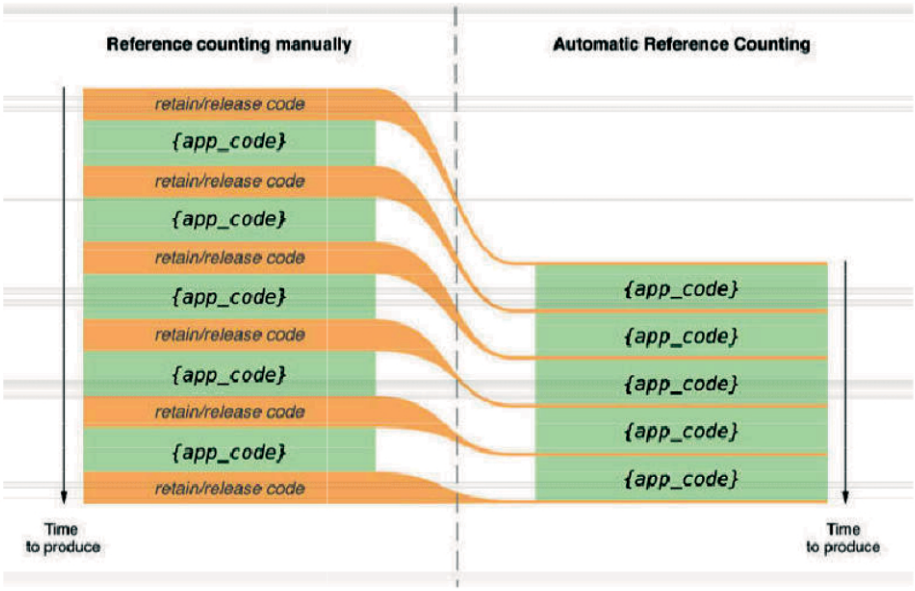

Fig3: Difference between Automatic and Manual Reference Counting

(Source: developer.apple.com)

Memory management using Reference counting is based on Ownership of object. An object can be owned by one or many owners. The object continues to exist till the time it has atleast one owner. The object is destroyed automatically at runtime if has no owner. Hence, **ownership** is the ability of causing object to be destroyed.

a. Reference Count Model

Memory is managed by maintaining life cycle of an object. When the objects are not needed, it should be de-allocated. When an object is created or copied, its retain count is set to 1. This retain count can increase with time if ownership interest is expressed by other objects. The retain count can also be decremented if objects relinquish their interest of ownership. The object will be maintained in memory till it's retain count is not zero. The object will be deallocated once it's retain count reaches zero.

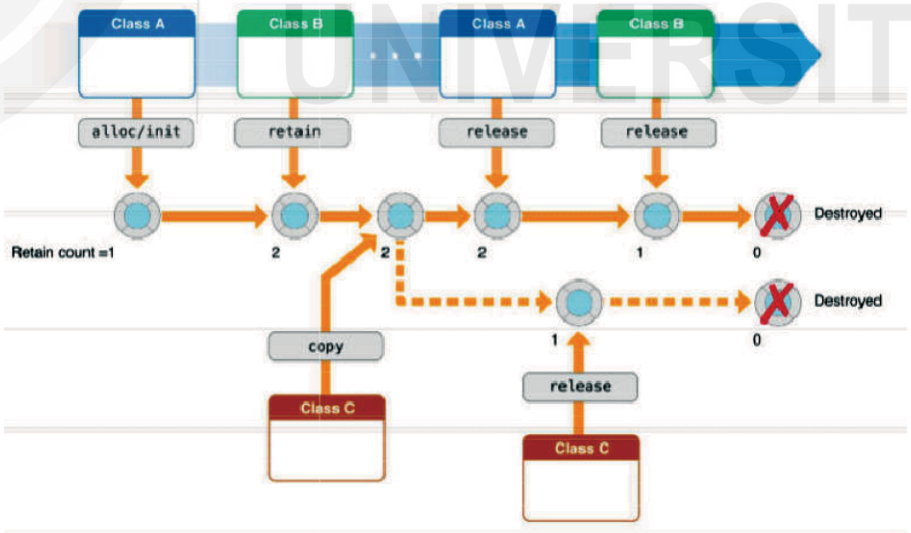

Fig 4: Reference Counting Model (Source: developer.apple.com)

b. Automatic Reference Counting

Automatic reference counting is a compiler provided feature that manages memory automatically of Object-C objects. Automatic reference counting is an ownership system which provides convention for management and transfer of ownership. It itself maintains

the object's lifetime requirements and user need not to remember about retain and releases. It allows object to be pointed by two types of references- strong and week. An object which is strongly referenced will be preserved and never deleted. The objects having back reference is called weak reference and can be de-allocated (destroyed). When no strong reference to objects is left, the object automatically becomes nil. Hence weak reference does not increase life time of objects. This model also imposes some restricts to prevent memory faults or issues.

4.7.2 Virtual Memory Management

Virtual memory overcomes the limitation of physical memory. For each process a logical address space is created by dividing the memory into pages. Apage table is maintained by Memory management units to map logical address to physical address. Logical address is provided to each process but if application wants to access a page with no physical address, page fault will occur. Virtual memory system is then responsible for invoking a special program called page fault handler. The page fault handler suspends the currently executing process, identifies free memory location and load the required page from the disk. The corresponding page table entry is then updated and can be used by returning control to the suspended process. This process of loading the page from disk is called Paging. There is no backing store in iOS hence pages cannot be moved out from primary memory (page write) hence, if no free space is available in primary memory than read only pages can be removed from memory which can be later loaded.

Page size in older versions of iOS is 4KB. While the newer version A9 supports 16KB pages.

Efficient use of memory is important for high performance of system. Reducing the amount of memory usage decreases memory footprint of applications and also reduces processor consumption. A fully integrated virtual memory is included with IOS which is always on. For 32- bit system it can provide upto 4 GB addressable space.

Data present on the disk (code pages) which is read only, can be removed from the primary memory and can later be loaded from the disk when needed, whereas, the iOS does not remove the writable data from memory. If the free memory reduces below a certain threshold, the running applications are asked to free up memory to make room for new applications or data voluntarily.

4.7.3 Page List in Kernel

Kernel maintains three lists of physical memory pages:

i. Active list - this list contains pages which are recently accessed

ii. Inactive list - this list contains pages which are not recently accessed but still present in physical memory.

iii. Free list - this list contains pages which are not currently allocated and is available for use by processes.

The pages that are not accessed are moved from active to inactive list by kernel. Also when the free memory reduces below a threshold, the pages that at unmodified and inactive are flushed by kernel and running applications are asked to free up memory by sending them low memory notification. Applications upon receiving notification remove strong references to as many objects as possible.

4.7.4 Page Fault

It is the process of loading the page into primary memory when needed by a process. This process is required as a result of page fault. The two types of fault that can occur are:

i. Soft fault - when the desired page is not mapped to the address space of process but it is present in physical memory.

ii. Hard fault - when the desired page is not present in the physical memory.

Whenever page fault occurs, kernel finds the VM object for accessed region and checks the resident list of VM object. If the required page is found in resident list, then soft fault is generated otherwise hard fault is generated. For handling soft fault, physical memory of page is mapped to the virtual memory and the page is marked as active by kernel. For handling hard fault, pager of VM object finds the page from disk and updates its map entry. The page after loading in primary memory is marked as active.

4.8 FILE MANAGEMENT SYSTEM

Storage of data, files, applications, operating system files is managed by file system. The default file system used in iOS is APFS for versions after iOS 10.3. Before this HFS+ was default file system. In iOS user cannot access file system directly but they can access it using applications.

4.8.1 iOS Directories

Applications can interact with file system only its sandbox directory. When the apps are installed, various container directories are created in its sandbox directory. Here every container is created for a specific role. There are two main containers –bundle container and data container. Data container is further divided into subdirectories used to organize app's data. In addition to theses containers, apps can also request for additional containers example iCloud container.

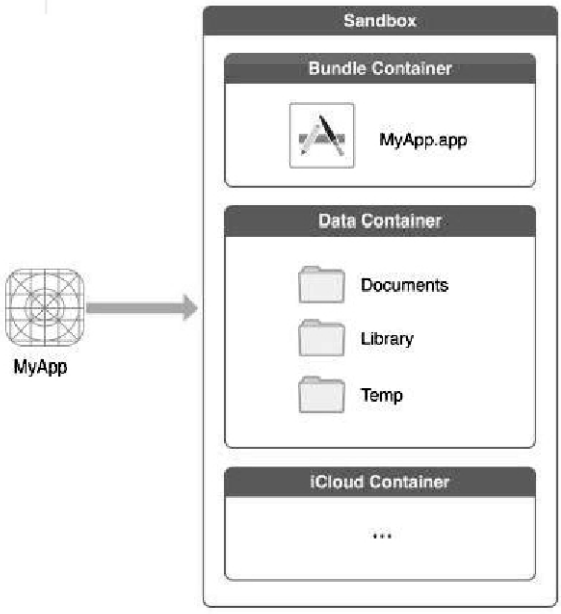

Fig5: Sandbox directory of iOSApp

(Source: developer.apple.com)

Files outside these containers are generally restricted for use by applications. Use of public system interfaces like contacts and music is exception to this rule.System framework in this case uses helper apps to perform file operations by modifying or reading from appropriate data storage locations.

Some of the commonly used directories in iOS sandbox are –

(i) Appname.app

This directory is created during app's installation. It contains the app's executable data and resources. User should not write to this directory because app may not launch due to change in signature. However, user can get read only access to app's bundle data.

(ii) Documents/

Data generated by user is stored in this directory. Files present in this directory can be accessed by user using file sharing. This directory is also backed up by iCloud and iTune.

(iii) Documents/Inbox

It is a subdirectory within Documents which is required by apps to open files asked by outside entities. Apps can read and delete data from this directory but cannot modify and create new files. Generally email attachments related to apps are placed in this directory by mail program. This directory is also backed up by iCloud and iTune.

(iv) Library/

This subdirectory is used to store data which is not required to be exposed to user. It contains applications support, preferences and cache subdirectories for iOS apps. Custom subdirectories can also be created by user but should not be used for user data.This directory is placed in data bundle. Except cache all the contents of this directory are also backed up.

(v) tmp/

This directory is used to store temporary file i.e. files that are not needed to persist. These file must be deleted by the apps when they are not needed. When apps are not running, system may delete theses files. Data in this directory is not backed up.

For proper organization of files, additional subdirectories can be created within Documents, tmp and Library by iOS apps.

4.8.2 iCloud Container

iCloud service allows user to store their documents, photos, files, etc. to be stored in cloud. Users when require can download these from their devices or share with other users. iOS allows users data be automatically backed up in cloud through this service. This feature is available in iOS 7 onwards.iCloud container is used to store files for apps that use iCloud. Files and folders can be created by apps in the same way as local file folders are created. Files and their attributes are copied to iCloud. Primary iCloud container of apps stores native files. Within each container two subdirectories – documents and data is present. Files present within document subdirectory are visible

to user as separate document which can be individually deleted. Whereas files not within document subdirectory is displayed as single entity in iCloud user interface called data.

4.8.3 Identification of File Type

To identify the type of file, two techniques are used in iOS-

(i) UTIs - Uniform Type Identifier: UTIs is a string that identifies class entitities uniquely. For all types of data and service it provides consistent identifiers. Not only files but they can also represent folders or directories in a flexible manner. They are used for identifying types of pasteboard. Some of the examples are:

- Public.jpeg - it represents a public type UTI identifying jpeg image data.
- com.apple.bundle - it represents an Apple type identifying bundle directory.
- com.apple.application-bundle - it represents an Apple type identifying bundled app.

(ii) Filename extension : It is a character string following filename and a period (.). Each extension represents a specific type of file. For example filename with .png represents an image file. Also period is a valid character for filename, hence only the string after last period is considered as extension to identify type.

4.8.4 Security of File System

(a) Sandbox environment

Applications during installation are automatically placed in a sandbox environment in iOS. Sandbox does not allows apps to write data into locations to which user should not write to. Every sandbox is created with few containers to which user can write. One app cannot write to containers of other apps or most directories outside their sandbox. Hence security of apps and data is maintained by restricting the access.

(b) File Access Control

File access control in iOS is maintained by two techniques – ACL& BSD. System assigns ACL and BSD permission to files created by apps.

i. Access Control List

a. Access control list is a list of control details which gives complete information of what operations can or cannot be done to a directory or file and by which users. Hence it allows different level of access writes to different user.

ii. BSD

a. In BSD instead of giving access writes to individual users, it allows access to be given on the basis of class of users. There exist three classes of users – owner of data or file, group of user, all other users.

(c) Encryption

iOS allows files on disk to be encrypted. Files that are needed to be encrypted can be designated by apps. When a device containing encrypted data is unlocked

by user, a decryption key is generated by system to enable access to file. Whereas, when user locks the device, to prevent unauthorized access, the decryption key is destroyed by the system.

☞ **Check Your Progress 1**

1) Describe the important features of iOS 14.4 version.

.....

.....

.....

.....

2) Describe the features of watchOS 7.3.

.....

.....

.....

.....

4.9 SUMMARY

In this unit we had studied iOS operating system. iOS is a mobile operating system developed by Apple Inc for iPhone. A description of how process management, memory management, file management, security in iOS is discussed in this unit. Along with various functions of the iOS operating system, its evolution, architecture, and other features were described.

In this course we had studied Operating Systems in general and 4 case studies.

4.10 SOLUTIONS / ANSWERS

Check Your Progress 1

1. Apple has released iOS 14.4 for compatible iPhone and iPod touch models. The new iOS update brings an updated Camera app that carries support for smaller QR codes. The latest software also introduces the option to classify Bluetooth devices in Settings and includes a few bug fixes. There are also patches for three vulnerabilities that might have been actively exploited. Apple iPad users have also received iPadOS 14.4 with a similar list of changes that is available for the iPhone users. Separately, Apple has released the HomePod software version 14.4 that brings the anticipated ultra-wideband (UWB) handoff feature specifically to the HomePod mini. The company has also rolled out watchOS 7.3 with the Apple Watch Unity watch face and tvOS 14.4 with security fixes and general stability improvements.

   The iOS 14.4 and iPadOS 14.4 updates bring a list of improvements to the compatible iPhone, iPod touch, and iPad models. One of those improvements is smaller QR codes support for the Camera app. This will enhance the existing QR

code recognition functionality of the default app. The latest software updates also include the option to classify Bluetooth device type in Settings to let users identify whether headphones or the built-in speaker will be used for audio notifications.

2. In January 2021, Apple has released watchOS 7.3 that brings the Unity watch face. The update also brings Time to Walk for Apple Fitness+ subscribers and expands the ECG app for the Apple Watch Series 4 and later to Japan, Mayotte, Philippines, and Thailand. The eligible Apple Watch users in Japan, Mayotte, Philippines, Taiwan, and Thailand will also get irregular heart rhythm notifications through the latest watchOS update. Furthermore, watchOS 7.3 fixes the issue that made the Control Centre and Notification Centre unresponsive when Zoom is enabled. The watchOS 7.3 update also patches the vulnerability tracked under CVE-2021-1782 that could be exploited by a bad actor.

4.11 FURTHER READINGS

1. John Ray, *iOS 9 Application Development in 24 Hours*, Sams Teach Yourself, 2019.
2. Jesse Feiler, *iOS App Development for Dummies*, Wiley, 2014
3. Matt Neuburg, *iOS 13 Programming Fundamentals with Swift*, 2019.
4. Rajiv Ramnath, *Beginnning iOS Programming for Dummies*, Wiley, 2014.
5. https://developer.apple.com/