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.

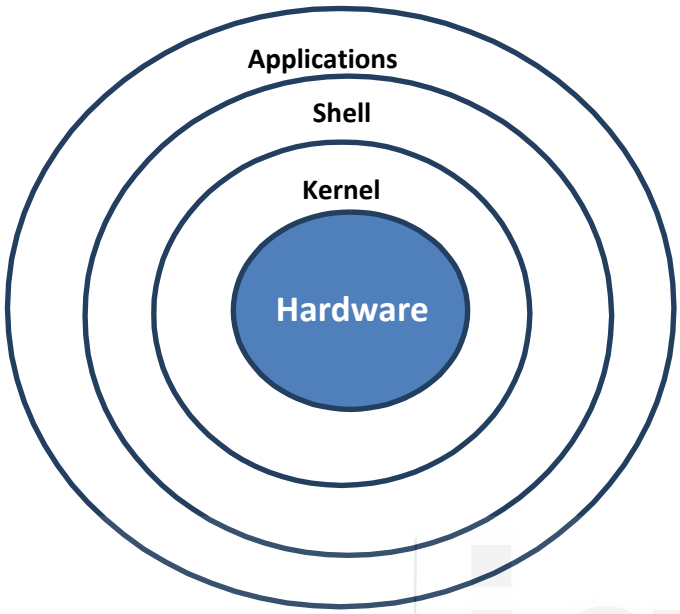

Fig 1: Linux Architecture

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.

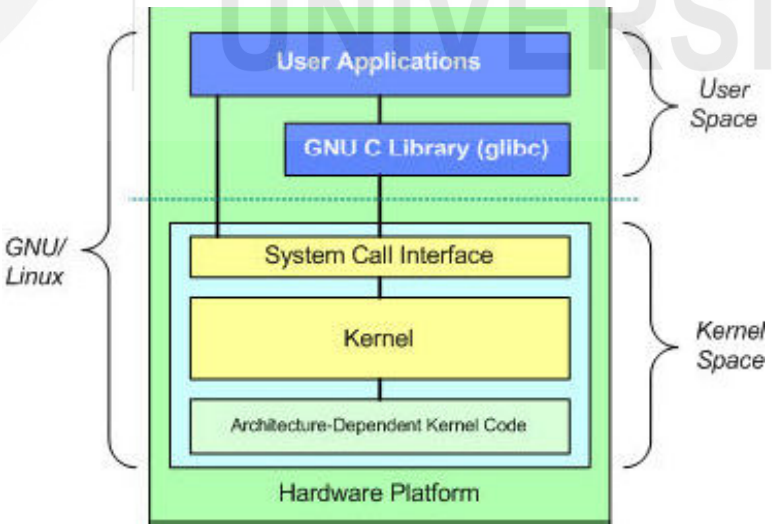

Fig. 2: Fundamental Architecture of Linux

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.

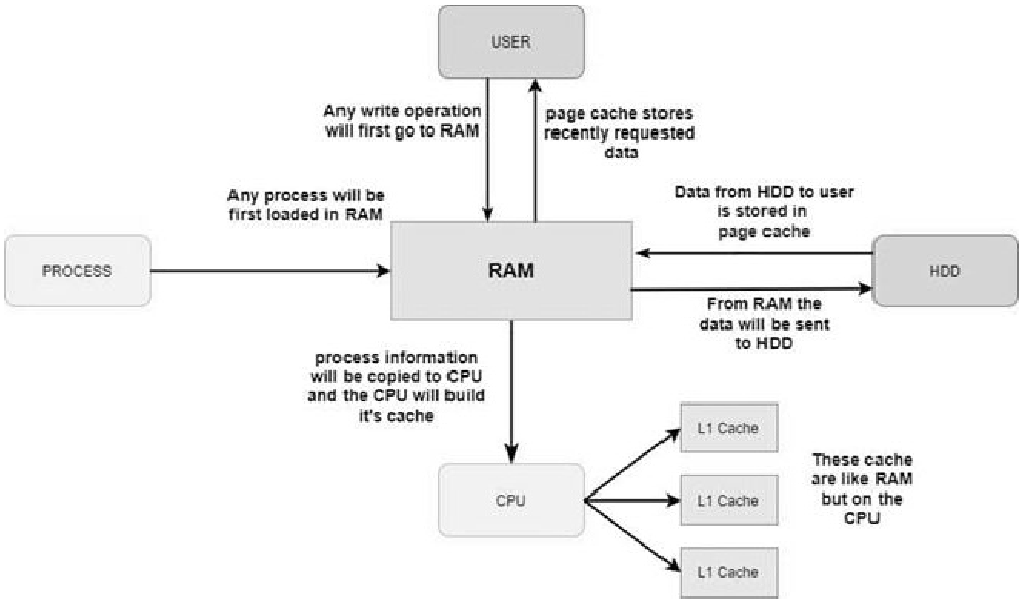

Figure 3: Memory Management in Linux

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.