UNIT-7

Linux Case Study

LINUX is one of popular version of UNIX Operating System. It is open source as its source code is freely available and it is free to use. Linux is one of the most reliable, secure and worry-free operating system available. Linux is generally far less vulnerable to ransomware, malware, or viruses. Linux has a number of different versions to suit any type of user. LINUX MINT, MANJARO, DEBIAN, UBUNTU, SOLUS etc. are popular linux distributions.

History: The History of LINUX began in the 1991 with the beginning of a personal project by a Finland student Linus Torvalds to create a new free operating system kernel. Since then, the resulting LINUX Kernel has been marked by constant growth throughout the history.

→ In the year 1991, LINUX was introduced by a Finland student Linus Torvalds.
→ In the year 1992, Hewlett Packard 9.0 was released.
→ In the year 1993, NetBSD 0.8 and FreeBSD 1.0 released.
→ In the year 1994, Red Hat LINUX was introduced.
→ In the year 1995, FreeBSD 2.0 and HP UX 10.0 were released.
→ In the year 1997, HP-UX 11.0 was released.
→ In the year 1998, the fifth generation of SGI Unix was released.
→ In the year 2001, Linus Torvalds released LINUX 2.4
→ In the year 2004, Ubuntu was released.
→ In the year 2006, Oracle released its own distribution of Red Hat.
→ In the year 2011, LINUX kernel 3.0 versions were released.
→ In the year 2013, Google LINUX based Android claimed 75% of the smartphone market share, in terms of no of phones shipped.
→ In the year 2014, Ubuntu claimed 22,000,000 users.

Kernel Modules: The LINUX kernel is a monolithic kernel i.e, it is one single large program where all the functional components of the kernel have access to all of its internal data structures and routines. A kernel module is an object file that contains code that can extend the kernel functionality at runtime. When a kernel module is no longer needed it can be unloaded. Most of

the device drivers are used in the form of kernel modules. Kernel modules are usually stored in the /lib/modules subdirectories. Kernel Modules includes following:

- Applications and OS services → These are the user application running on the LINUX system. OS services include utilities and services like shells, libraries, compilers etc.
- LINUX Kernel → Kernel abstracts the hardware to the upper layers. It mediates and controls access to system resources.
- Hardware → This layer consists of the physical resources of the system that finally do the actual work. It includes the CPU, the hard disk, system RAM etc.

⊛. Process Management:

A process refers to a program in execution; it's a running instance of a program. A process generally takes an input, processes it and gives the appropriate output. Tuning or controlling a process is called Process Management. In LINUX two vectors define a process: argument vector and environment vector. The argument vector has the command line arguments used by the process. The environment vector has a (name, value) list where different environment variable values are specified. There are two types of processes in Linux:

- Foreground process → By default, all the processes run in the foreground. When a process is run in foreground, no other process can be run on the same terminal until the process is finished or killed. When issuing this type of process, the system receives input from the keyboard (stdin) and gives output to the screen (stdout).
- Background process → Adding '&' to a foreground command makes it a background process. A background process runs on its own without input from the keyboard (stdin) and waits for input from the keyboard. While the process runs in the background, other processes can be run in the foreground. The background process will be in stop state till input from the keyboard

is given (usually 'Enter' key), then becomes a foreground process and gets executed. Only after the background process becomes a foreground process, that process gets completed else it will be a stop state.

⊛ **Linux Scheduling:-** Linux scheduling is based on the time sharing technique: several processes run in "time multiplexing" because the CPU time is divided into slices, one for each runnable process. If a currently running process is not terminated when its time slice or quantum expires, a process switch may take place. Time sharing relies on timer interrupts and thus transparent to processes. No additional code needs to be inserted in the programs to ensure CPU time sharing. The schedular always succeeds in finding a process to be executed. Every Linux process is always scheduled according to one of the following scheduling classes;

SCHED_FIFO → It is a First-In, First-Out real-time process. When the schedular assigns the CPU to the process, it leaves the process descriptor in its current position in the run queue list. If no other higher-priority real-time process is runnable, the process continues to use the CPU as long as it wishes, even if other real-time processes that have the same priority are runnable.

SCHED_RR → It is a Round Robin real-time process. When the schedular assigns the CPU to the process, it puts the process descriptor at the end of the run queue list. This policy ensures a fair assignment of CPU time to all SCHED_RR real-time processes that have the same priority.

SCHED_NORMAL → It is a conventional, time-shared process.

⊛. Inter-process Communication:

Inter-Process Communication (IPC) refers to a mechanism, where the operating systems allow various processes to communicate with each other. This involves synchronizing their actions and managing shared data. For inter-process communication LINUX has three main components:

i) **Module Management** → For new modules this is done at two levels- the management of kernel refrenced symbols and the management of the code in kernel memory. The LINUX kernel maintains a symbol table and symbols defined here can be exported explicitly. The module management system also defines all the required communication interfaces for newly inserted module. With this done, processes can request the services from this module.

ii) **Driver registration** → Usually the registeration of drivers is maintained in a registeration table of the module. The registeration of drivers contains the following:

→ Driver context identification as a bulk device or network driver.
→ File system context to store files in LINUX virtual file system or network file system like NFS.
→ Network protocols and packet filtering rules.
→ File formats for executable and other files.

iii) **Conflict Resolution** → The PC hardware configuration is supported by a large number of chip set configurations and with a large range of drivers for SCSI devices, video display devices and adaptors, network cards. This results in the situation where we have module device drivers which vary over a very wide range of capibilities and options. This necessitates a conflict resolution mechanism to resolve accesses in a variety of conflicting concurrent accesses. The conflict resolution mechanisms help in preventing modules from having an access conflict.

⊛. Memory Management:-

Following are the two major components in LINUX memory management:

1. **The page management** → Pages are usually of a size which is a power of 2. LINUX allocates a group of pages using a buddy system. "Page allocator" software is responsible for both allocation, as well as freeing the memory. The basic memory allocator uses a buddy heap which allocates contiguous are of size 2^n > the required memory with minimum n obtained by successive generation of "buddies" of equal size.

Example: Suppose we need memory of size 1556 words. Starting with a memory size 16K we would precede as follows:

→ First create 2 buddies of size 8K from the given memory size i.e, 16K
→ From one of the 8K buddy create two buddies of size 4K each.
→ From one of the 4K buddy create two buddies of size 2K each.
→ Use one of the most recently generated buddies to accommodate the 1556 size memory requirement.

i.e, 2000 word size

#Note that for a requirement of 1556 words, memory chunk of size 2K words satisfies the property of being the smallest chunk larger than the required size.

2. **Virtual Memory Management** → Linux supports virtual memory, that is, using a disk as an extension of RAM so that the effective size of the usable memory grows correspondingly. The kernel will write the contents of a currently unused block of memory to the hard disk so that the memory can be used for another purpose. When the original contents are needed again, they are read back into memory. Reading and writing the hard disk is slower than the using real memory, so programs don't run as fast it should. The part of hard disk that is used as virtual memory is called the swap space. Linux can also use normal file system or a seperate partition for swap space. A swap partition is faster, so we can go for swap partition if needed.

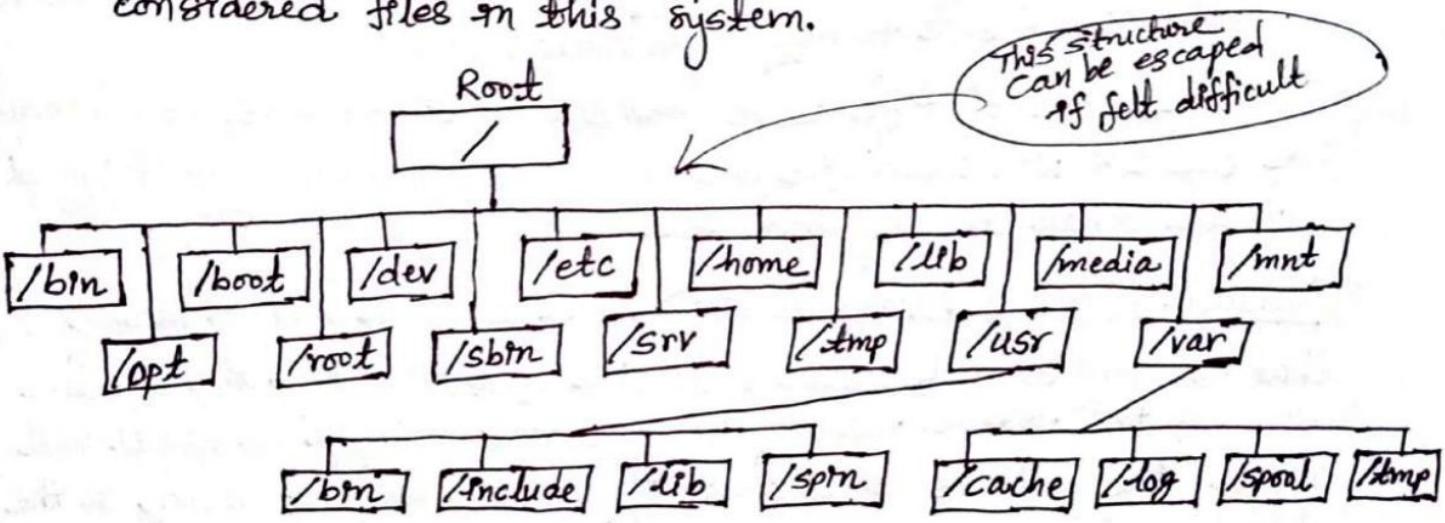
⊛. File System Management:

Linux file system management refers to how linux-based computers organize, store and track system files. The file system is basically a combination of directories or folders that serve as a placeholder for addresses of other files. There is no distinction between a file and a directory in Linux file system, because a directory is considered to be a file containing names of other files. In Linux, all the files and directories are located in a tree-like structure. Linux has three types of files:

i) Regular Files → It includes files like text files, images, binary files etc. Such files can be created using the touch command.

ii) Directories → Windows call these directories as folders. The root directory (/) is the base of the system. We could create new directories with mkdir command.

iii) Special Files → It includes physical devices such as a printer which is used for I/O operations. I/O devices are also considered files in this system.



⇒ The list below provides a short overview of the most important high-level directories on a Linux system.

Directory		Contents
/	→	Root directory which is the starting point of the directory tree.
/bin	→	Essential binary files, such as commands that are needed by both the system administrator and normal users.

/boot → Static files of the boot loader.
/dev → Files needed to access host specific devices.
/etc → Host-specific configuration files.
/lib → Essential shared libraries and kernel modules.
/media → Mount points for removable media.
/mnt → Mount point for temporarily mounting a file system.
/opt → Add-on application software packages.
/root → Home directory for the superuser root.
/sbin → Essential system binaries.
/srv → Data for services provided by the system.
/tmp → temporary files.
/usr → Secondry hierarchy with read-only data.
/var → Variable data such as log files.

⊛. Device Management:

Linux device management includes the management of I/O and other hardware devices. Modern Linux distributions are capable of identifying a hardware component which is plugged into an already-running system. There are a lot of user-friendly distributions like Ubuntu, which will automatically run specific applications like Rhythmbox when a portable device like an iPod is plugged into the system.

The process of inserting devices into a running system is achieved in a Linux distribution by a combination of three components:- Udev, HAL and Dbus. Udev creates or removes the device node files in the /dev directory as they are plugged in or taken out. The HAL gets information from the Udev ~~device~~ service, when a device is attached to the system and it creates a XML representation of that device. Dbus is like a system bus which is used for inter-process communication.

If my notes really helped you,then you can support me on esewa for my hardwork.

Esewa ID: 9806470952