Unit 7 : Linux Case Study

- Ankit Pangeri

DATE

LINUX is one of the popular versions of UNIX operating system. It is open source as its source code is freely available. It is free to use. It was designed considering UNIX Compatibility. It was designed to offer a free or low cost OS for personal computer users. It gained reputation as a fast performing & very efficient system.

* History

- 1991 :- LINUX was introduced by a Finland Student Linus Torvalds
- 1992 :- Hewlett Packard UNIX (Hp-UX) 9.0 was released
- 1993 :- NetBSD 0.8 and FreeBSD 1.0 released
- 1994 :- Red hat LINUX was introduced.
- 1995 :- FreeBSD 2.0 & Hp UX 10.0 were released
- 1996 :- K Desktop Environment was developed by Matthias Ettrich
- 1997 :- Hp-UX 11.0 was released.
- 1998 :- 5th Gen of SGI unix & FreeBSD 3.0 was released
- 2000 :- Agreement of Caldera systems with SCO Server Software division & professional service division.
- 2001 :- Linus released LINUX 2.4 version source code
- 2004 :- Windows name was changed to Winspire.
- 2004 :- 1st release of Ubuntu was released.
- 2005 :- project openSuSE began a free distribution
- 2006 :- Oracle released its own distribution of Red Hat.
- 2007 :- Dell started distributing laptops with Ubuntu preinstalled
- 2011 :- LINUX Kernel 3.0 versions were released
- 2013 :- Google linux based Android claimed 75% of smartphone market share
- 2014 :- Ubuntu claimed 22,000,000 users

DATE

* Features of LINUX OS

→ Portable
→ Open Source
→ Multiuser
→ Multiprogramming
→ Hierarchical File system
→ Shell
→ Security.

* Components of Linux System

Linux OS has primarily three components

a) Kernel : It is a core part of linux. It is responsible for all major activities of this OS. It consists of various modules & it interacts directly with the underlying hardware. It provides the required abstraction to hide low level hardware details to system or application programs

b) System library: They are special functions or programs which application programs or system utilities access kernel's features. These libraries implement most of the functionalities of os & do not require kernel module's code access rights.

c) System Utility: System utility programs are responsible to do specialized, individual level tasks.

PAGE

DATE

* Kernel Modules

The LINUX Kernel is a monothalic kernel, ie, it is one single large program where all the functional components of the kernel have access to all of its internal data structures & routines. A kernel module is an object file that contains code that 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 include following:-

i) Application and OS Services. → These are the user application running on the LINUX system. OS services include utilities & services like shells, libraries, compilers, etc.

ii) LINUX kernel:→ Kernel abstracts the hardware to the upper layers. It mediates and controls access to the system resources.

iii) 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.

PAGE

DATE

* Process Management

A process refers to a program in execution; it's a running instance of a program. It is made up of the program instruction, data read from files, other programs or inputs from a system user.

Whenever you issue a command in Unix, it creates or starts a new process. The OS tracks processes through a five-digit ID number known as the pid or the process ID. Each process in the system has unique pid. In Linux, internally, both processes & threads have the same kind of representation. LINUX processes & threads are POSIX compliant & are supported by a threads library package which provides for two kinds of threads: user & kernel. User-Controlled scheduling can be used for user threads. Kernel threads are scheduled by the kernel.

Types of processes:

When you start a process, there are two ways you can run it:

(Interactive)

1) Foreground Processes: By default, every program that you start runs in the foreground. It gets its input from the keyboard & sends its output to the screen. When a process is running in foreground, no other process can be run on the same terminal until the process is finished or killed.

PAGE

(non-interactive/ automatic)

2) Background processes

Adding '&' to a foreground command makes it a background process. It runs on its own without input from the keyboard. While the process runs in background, other process 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 & gets executed. Only after the background process becomes a foreground process, it gets completed else it will be a stop state.

* Creation of a process in Linux

A new process is normally created when an existing process makes an exact copy of itself in memory. The child process will have the same environment as its parent, but only the process ID no. is different. Two ways for creating process

- Using The system() Function → this method is relatively simple, however, it's inefficient & has significantly certain security risks.
- Using fork() and exec() function → It is little advanced but offers greater flexibility, speed, together with security

process states in linux : Same as previous unit

DATE

* 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 process. No additional code needs to be inserted in the programs to ensure cpu time sharing. The scheduler always succeeds in finding a process to be executed. Every linux process is scheduled according to one of the following scheduling classes:-

- SCHED_FIFO → It is a first-In, First-Out real-time process. When a scheduler assigns the cpu to a process, it leaves the process descriptor in it's 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 same priority are runnable.

- SCHED-RR → It is a Round Robin real time process. When the scheduler 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.

PAGE

DATE

¥ Inter-process Communication (IPC)

IPC refers to a mechanism where the OS allows various processes to communicate with each other. This involves synchronizing their actions and managing shared data. For IPC, LINUX has three main components:-

i) Module Management :- For new modules, this is done at two levels. the management of kernel referenced symbols & the management of code in the 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, process can request the services from this module.

ii) Driver registration :- Usually the registration of drivers is maintained in a registration table of the module. The registration of drivers contains the following
→ Driver context identification as bulk device or network driver
→ File system context to store files in LINUX virtual filesystem
→ Network protocols and packet filtering rules
→ File formats for executable & other files

iii) Conflict Resolution :- The PC hardware configuration is supported by a large no. of chipset configs & with a large range of drivers for SCSI devices, video display devices & adaptors which results in conflict. This makes it necessary to use a conflict resolution mechanism to resolve access in a variety of conflicting concurrent access. It helps in

DATE

preventing modules from having an access conflict.

Memory Management.

The two major components in LINUX memory management are:

- (The page management) :-

Pages are usually of a size which is a power of 2. Given the main memory, LINUX allocates a group of pages using a buddy system. The allocation as well as freeing of memory is the responsibility of a software called "page allocator". The basic memory allocator uses a buddy heap which allocates contiguous area of size 2^n > the required memory with minimum n obtained by successive generation of "buddies" of equal size.

- (Virtual Memory Management) :-

Linux supports virtual memory, ie. 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. Read & write on the hard disk is slower, so programs don't run as fast as they should. The part of hard disk used as virtual memory is called swap space. Linux can also use normal file system or a separate partition for swap space, which is faster.

PAGE

classmate

DATE

* File system management Approaches

Linux filesystems refer 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 address of other files.

In Linux, all files & directories are located in a tree like structure. The topmost directory is referred to as the file system root or just /. The counterpart of / in a windows system would probably be (C:). All other directories in Linux can be accessed from the root directory & are arranged in a hierarchical structure.

The following table provides a short overview of the most important higher-level directories on a Linux system.

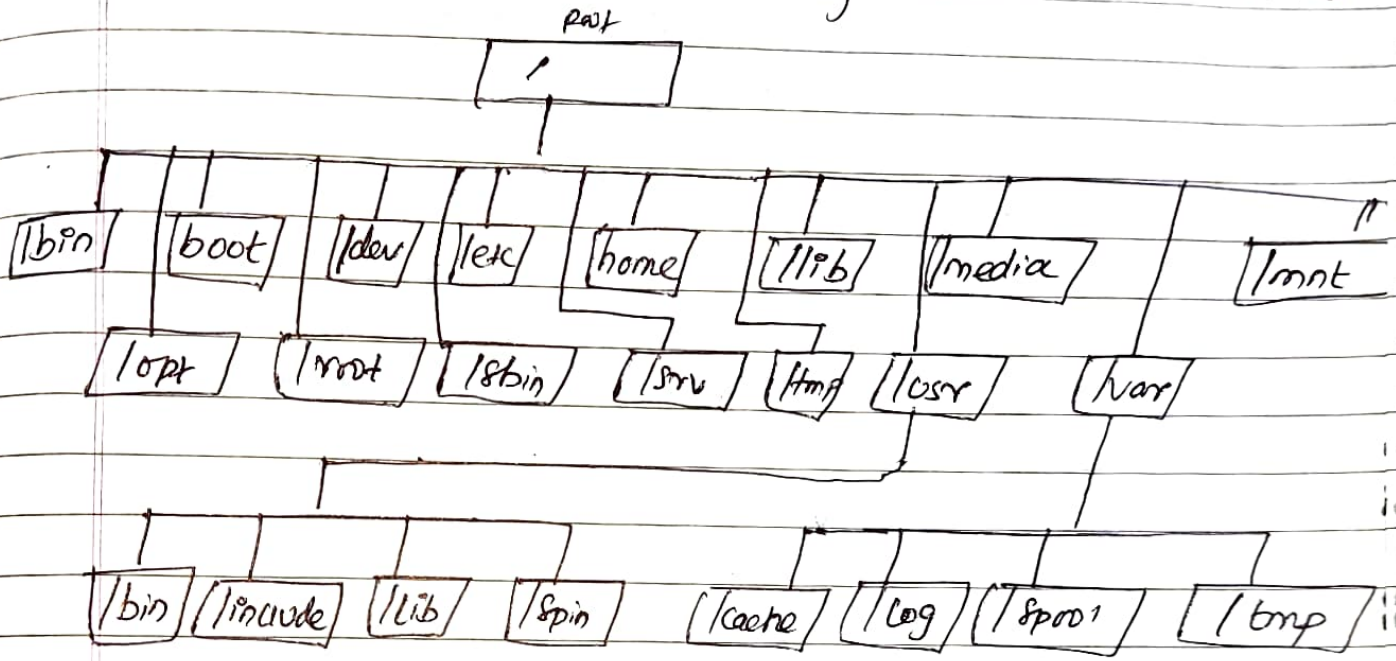
Directory	Content
/	Root directory - starting point of the tree
/bin	Essential binary files
/boot	Static files of the boot loader
/dev	files needed to access host specific devices
/etc	Host-specific system config. files
/lib	essential shared libraries & kernel modules
/media	mount points for removable media
/mnt	mount points 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

classmate

PAGE

DATE

/tmp	Temporary files.
/usr	Secondary hierarchy with read-only data.
/var	Variable data such as log files.
/windows	Available when you dual boot windows.



* Device Management Approaches

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, mint, which are automatically run specific applications like Rythmbox 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

DATE

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 service, when a device is attached to the system & it creates a XML representation of that device. Dbus is like a system bus which is used for inter-process communication.

Questions asked from this Chapter

Q. Write short notes on (2078 - 2.5 marks)
- IPC in linux

Q. Write short notes on (2076 - 2.5 marks)
- Linux file system

Q. What is Linux kernel? (imp)
Q. What are the basic components of linux? (imp)
Q. Explain device management in Linux. (imp)
Q. Explain memory management in Linux. (imp)
Q. What are the scheduling classes in Linux? (imp)
Q. What are kernel modules in linux? (imp)
Q. Explain process management in Linux. (imp)