LINUX SYSTEM

Unit Structure

- Objectives
- Linux introduction and File System
- Basic Features
- Advantages
- Basic Architecture of UNIX/Linux System
- Summary
- Model Question

Objective

- To explore the history of the UNIX operating system from which Linux is derived and the principles which Linux is designed upon.
- To examine the Linux process model and illustrate how Linux schedules processes and provides inter-process communication.
- To explore how Linux implements file systems and manages I/O devices.

Linux History

Linux looks like UNIX system. Linux is developed by Linux Torvalds. Linux development began in 1991. Linux source code is available for free on the internet. The basic Linux system is standard environment for applications and user programming.

Linux development revolved largely around the central operating-system kernel-the core, privileged executive that manages all system resources and that interacts directly with the computer hardware. It is useful to make the distinction between the Linux kernel and a Linux system. The Linux Kernel is an entirely original piece of software developed from scratch by the Linux community. The Linux system, includes *a* multitude of components, some written from scratch, others borrowed from other development projects, and still others created in collaboration with other teams.

The basic Linux system is a standard environment for applications and user programming, but it does not enforce any standard means of managing the available functionality as a whole. A Linux distribution includes all the standard components of the Linux system, plus *a* set of administrative tools to simplify the initial installation and subsequent upgrading of Linux and to manage installation and removal of other packages on the system. A modern distribution also typically includes tools for management of file systems, creation and management of user accounts, administration of networks, Web browsers, word processors, and so on.

The Linux Kernel

The first Linux kernel released to the public was Version 0.01, dated May 14, 1991. It had no networking, ran only on 80386-compatible Intel processors and PC hardware and had extremely limited device-driver support. The only file system supported was the Minix file system- the first Linux kernels were cross-developed on *a* Minix platform. However, the kernel did implement proper UNIX processes with protected address spaces.
The next milestone version, Linux 1.0, was released on March 14, 1994.

The single biggest new feature was networking: 1.0 included support for UNIX's standard TCP-*iP* networking protocols, as well as a BSD-compatible socket interface for networking programming. Device-driver support was added for running IP over an Ethernet or over serial lines or modems.

The 1.0 kernel also included a new, much enhanced file system without the limitations of the original Minix file system and supported a range of SCSI controllers for high-performance disk access. The developers extended the virtual memory subsystem to support paging to swap files and memory mapping of arbitrary files.

System V UNIX-style interprocess communication (IPC) including shared memory, semaphores, and message queues, was implemented.

Development started on the 1.1 kernel stream, but numerous bug-fix patches were released subsequently against 1.0. A pattern was adopted as the standard numbering convention for Linux kernels. Kernels with an odd minor-version number, such as 1.1, 1.3, and2.1, are even numbered minor-version numbers are stable Updates against the stable kernels are intended only as remedial versions, whereas the development kernels may include newer and relatively untested functionality.

Linux 2.0 : The memory-management code was substantially improved to provide a unified cache for file-system data independent of the caching of block devices. As a result of this change, the kernel offered greatly increased file-system and virtual memory performance.File-system caching was extended to networked file systems and writable memory-mapped regions also were supported.

The 2.0 kernel included much improved TCP /IP performance and a number of new networking protocols were added, including Apple Talk, AX.25 amateur radio networking, and ISDN support. The ability to mount remote netware and SMB (Microsoft Lan Manager) network volumes was added.

Linux 2.2 : Improvements continued with the release of Linux 2.2 in January 1999. A port for Ultra SPARC systems was added. Networking was enhanced with more flexible firewalling, better routing and traffic management, and support for TCP large window and selective acks. Signal handling, interrupts, and some I/0 were locked at a finer level than before to improve symmetric multiprocessor (SMP) performance.

Advances in the 2.4 and 2.6 releases of the kernel include increased support for SMP systems, journaling file systems, and enhancements to the memory management system. The process scheduler was modified in Version 2.6, providing an efficient 0(1) scheduling algorithm. In addition, the LimiX 2.6 kernel is now preemptive, allowing a process to be preempted while running in kernel mode.

The Linux **System :**
Linux kernel forms the core of the Linux project, but other components make up the complete Linux operating system. Linux kernel is composed entirely of code written from scratch specifically for the Linux project, much of the supporting software that makes up the Linux system is not exclusive to Linux but is common to a number of UNIX-like operating systems. In

particular, Linux uses many tools developed as part of Berkeley's BSD operating system, MIT's X Window System, and the Free Software Foundation's GNU project.

The Linux system as a whole is maintained by a loose network of developers collaborating over the Internet, with small groups or individuals having responsibility for maintaining the integrity of specific components. A small number of public Internet file-transfer-protocol (FTP) archive sites act as de facto standard repositories for these components.

The File System Hierarchy structure document is also maintained by the Linux community as a means of ensuring compatibility across the various system components. This standard specifies the overall layout of a standard Linux file system; it determines under which directory names configuration files, libraries, system binaries, and run-time data files should be stored.

Linux Distribution

- Standard, precompiled sets of packages, or *distributions*, include the basic Linux system, system installation and management utilities, and ready-to-install packages of common UNIX tools
- The first distributions managed these packages by simply providing a means of unpacking all the files into the appropriate places; modern distributions include advanced package management
- Early distributions included SLS and Slackware
- *Red Hat* and *Debian* are popular distributions from commercial and noncommercial sources, respectively
- The RPM Package file format permits compatibility among the various Linux distributions

Linux Licensing

- The Linux kernel is distributed under the GNU General Public License (GPL), the terms of which are set out by the Free Software Foundation
- Anyone using Linux, or creating their own derivative of Linux, may not make the derived product proprietary; software released under the GPL may not be redistributed as a binary-only product

Design Principles

- Linux is a multiuser, multitasking system with a full set of UNIX-compatible tools
- Its file system adheres to traditional UNIX semantics, and it fully implements the standard UNIX networking model
- Main design goals are speed, efficiency, and standardization
- Linux is designed to be compliant with the relevant POSIX documents; at least two Linux distributions have achieved official POSIX certification
- The Linux programming interface adheres to the SVR4 UNIX semantics, rather than to BSD behavior

Components of a Linux System

Any Linux system consists of three components.

1. Kernel
2. System libraries
3. System utilities

Kernel:
The Kernel is responsible for maintaining all the important abstraction of the operating system. It includes virtual memory and processes. Kernel is most important component of the Linux system.

System libraries:
It contains standard set of functions through which application can interact with the Kernel. System library implements much of the operating system functionality that does not need the full privileges of Kernel code.

System utilities:
System utilities are programs that perform individual, specialized management tasks.
Fig.15.1 shows the components of the Linux system.

All the Kernel code executes in the processors privileged mode with full access to all the physical resources of the computer. Linux refers to this privileged mode as Kernel mode. In Linux, no user mode code is built into the Kernel.

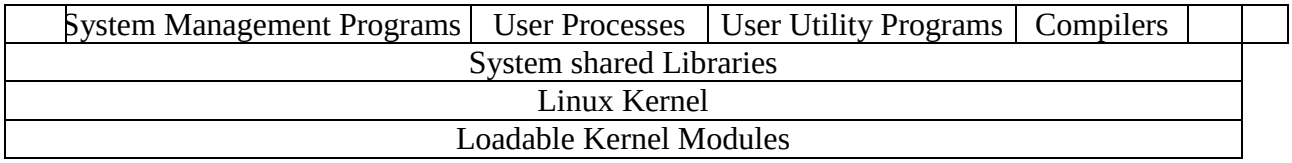

Fig: 15.1 Components of the Linux System.

- Any operating system support code that does not need to urn in Kernel mode is placed into the system libraries instead.
- The Kernel is created as a single, monolithic binary. All Kernel code and data structures are kept in a single address space, no context switches are necessary when a process calls an operating system function. Core scheduling, virtual memory code also occupied same address space.
- Linux Kernel provides all the functionality necessary to run processes, and it provides system services to give arbitrated and protected access to hardware resources.
- The Linux system includes a wide variety of user mode programs both system utilities and user utilities.

Kernel Modules

- Kernel code executes in Kernel mode with full access to all the physical resources of the computer. Sections of Kernel code that can be compiled, loaded and unloaded independent of the rest of the Kernel
- A Kernel module may typically implement a device driver, a file system, or a networking protocol. The module interface allows third parties to write and distribute, on their own teams, device drivers or file systems that could not be distributed under the GPL.
- The loadable Kernel modules run in privileged Kernel mode. Kernel modules allow a Linux system to be setup with a standard, minimal Kernel, without any extra device drivers built in. Kernel management allows modules to be dynamically loaded into memory when needed.

- Linux Kernel modules has three components.
  1. Module management
  2. Driver registration
  3. Conflict resolution mechanism

1. **Module management**

1. It support loading modules into memory and letting them talk to the rest of the Kernel.
2. Linux maintains an internal symbol table in the Kernel.
3. Module loading is split into two separate sections:
   - a. Managing sections of module code in Kernel memory
   - b. Handling symbols that modules are allowed to reference.
4. Symbol table does not contain the full set of symbol defined in the Kernel during latter compilation
5. The module requestor manages loading requested, but currently unloaded modules.
6. It also regularly queries the Kernel to see whether a dynamically loaded module is still in use and will unload it when it is no longer actively needed
7. Original service request will complete once the module is loaded.

2. Driver Registration.

1. Allows modules to tell the rest of the kernel that a new driver has become available.
2. Kernel maintains dynamic tables of all known drivers.
3. Kernel also provides a set of routines to allow drivers to be added or removed.
4. Registration table contains following:
   a. Device drivers
   b. File systems
   c. Network protocols
   d. Binary format.
5. Device drivers may be block or character devices.
6. Files system contains network file system, virtual file system etc.
7. Network protocol includes IPX, packet filtering rules for a network.

3. Conflict Resolution

1. A mechanisms that allows different device drivers to reserve hardware resources and to protect those resources from accidental use by another driver.
2. Linux provides a ventral conflict resolution mechanism.
3. Conflict resolution module aims are
   a. To prevent modules from clashing over access to hardware resources.
   b. Prevent auto probes from interfering with existing device drivers.
   c. Resolve conflicts among multiple drivers trying to access the same hardware.
4. Kernel maintains lists of allocated hardware resources.

Basic Features

UNIX has a number of features, mostly good and some bad, but it is necessary to know at least some of them:

1. UNIX is Portable: Portability is the ability of the software that operates on one machine to operate on another, different machine. In UNIX there are two types of portability, that of the UNIX operating system itself (i.e. the Kernel program) and of the application program.

Advantages of portability are:

a) Portable application program decrease the programming costs.

b) Retraining is avoided as the end user works on a similar system with enhanced capabilities.

2. Open system: The system that more than one system can access the same data at the same time. Several people involved in a common project can conveniently access each other's data. Apart from data, other resources like memory (RAM), the CPU (the chip), or the hard disk can be used by many users simultaneously.

3. Multi-user Capability: This means that more than one system can access the same data at the same time. Several people involved in a common project can conveniently access each other's data. Apart from data, other resources like memory (RAM), the CPU(the chip), or the hard disk can be used by many users simultaneously.

4. Multi-tasking Capability: Multi-tasking means that a user can perform more than one tasks at the same time. A user can do this by placing some tasks in the background while he works on a task in the foreground. An ampersand (&) placed at the end of the command line sends the process in background.

5. Hierarchical File System: Hierarchical structure offers maximum flexibility for grouping information in a way that reflects in natural structure. The programs and data can be organized conveniently since files can be grouped according to usage.

6. The shell: User interaction with UNIX is controlled by the shell, a powerful command interpreter. The shell has various capabilities like redirecting the application input and output and also to progress a group of files with a single command.

7. UNIX has built in networking: The UNIX has various built in programs and utilities like UUCP, mail, write, etc. With these utilities one can communicate with other user or one server to another.

8. Security: Computer system security means protecting hardware and the information contained within the system. Security means to avoid:

a) Unauthorized access to the computer system.

b) Unauthorized examination of output.

c) Unauthorized tapping of data.

d) Destruction of software data by mistake or on purpose.

e) Examination of sensitive data by unauthorized users and alteration of sensitive data without detection.

The UNIX provides the features like:

a) Password protection for system access.

b) Control access to individual files.

c) Encryption of data files.

9. Software Development Tools: UNIX offers an excellent variety of tools for software development for all phases, from program editing to maintenance of software.

Basic Architecture of UNIX/Linux System

Fig Architecture of UNIX System

The figure shows the architecture of the UNIX system in the form of various layers. The inner most layer i.e. core is the hardware of the computer system. The kernel is a software which directly comes in contact with the hardware and controls the hardware. A layer above this is the

shell. Above this the various utilities like nroff, troff, vi, as, etc. and the various compilers and other application program reside. Let us see these layers and their functions in detail:

1. Hardware: The hardware at the centre of the structure provides the operating system with basic services. The hardware constitutes all the peripherals like memory (RAM, HDD, FDD, CD etc), processor, mouse and other input devices, terminal (i.e. VDUs), printers etc.

2. The Kernel: The operating system interacts directly with the hardware, providing common services to programs and insulating user program complexities of hardware. The operating system is also called as Kernel. This kernel interacts directly with the hardware. Because of this hardware isolation of user program, it is easy to move the user programs easily between UNIX systems running on different hardware.

3. Shell: Shell is actually the interface between the user and the Kernel that isolates the user from knowledge of Kernel functions. The shell accepts the commands keyed in by the users and checks for their syntax and gives out error messages if something goes. These command after getting interpreted by the shell are provided to kernel for appropriate action. It also provides the features of pipe (|) and redirection (i.e. <, >, >>, <<). The shell also has a programming capability of its own. There are a number of shells available in various UNIX flavours but common are-

a) Bourne Shell (sh)
b) C Shell (csh)
c) Korn Shell (ksh)

The other shells are: bash, restricted Shell(rsh) and visual Shell(vsh). The bourne Shell (sh) is also called a standard shell.

4. UNIX utilities: The UNIX system Kernel does only level jobs necessary to schedule processes, keep track of files, and control hardware devices.

All other operating system functions are done by utility programs written as software tools.
Following are some of the utilities that play a prominent role in the operating of the UNIX system.
a) **Init** initializes the process creation mechanism.

For each terminal where a login is allowed, init creates a process that prints the login: message.
b) **Getty** process ―conditions‖ the connection to the terminal so that the terminal can communicate with the computer and then prints the login: message.

c) **Login** when an user responds to the login prompt the login program replaces getty. If the user account has a password, login prints the password : message and checks to see that the password entered is correct.

If correct, program named in password file, usually Bourne shell or C shell replaces login and the user is successfully logged into the system.
d) **stty** changes a terminal characteristics.
e) **mkfs** builds a file system.
f) **mknod** builds a special file system.
g) **mount unmount** mounts and unmounts a file system.
h) **syne** writes a disk block images from memory to disk.

Summary

Linux is a modern, free operating system based on UNIX standards. It has designed to run efficiently and reliably on common PC hardware; it also runs on a variety of other platforms. It provide a programming interface and user interface compatible with standard UNIX systems and can run a large number of UNIX applications, including an increasing number of commercially supported applications.
Linux has not evolved in a vacuum a completely Linux system includes many components that were developed independently of Linux. The core Linux operating system kernel is entirely original, but allows much existing free UNIX software to run, resulting in a entire UNIX compatible operating system free from proprietary code.
The Linux kernel is implemented as a traditional monolithic kernel for performance reasons, but it is modular enough in design to allow most drivers to be dynamically loaded and unloaded at run time.

Model Questions

Q.1 List different components of a Linux system?
Q.2 What are Kernel modules?
Q.3 Explain Basic architecture of UNIX/Linux system?
Q.4 Explain basic features of UNIX/Linux?
Q.5 Define and explain Shell?