UNIT 4: LINUX KERNEL

UNIT STRUCTURE

4.1 Learning Objectives

4.2 Introduction

4.3 Linux Kernel Concepts

- 4.3.1 Major Functions of Linux Kernel
- 4.3.2 Architecture of Linux Kernel
- 4.3.3 Features of Linux Kernel

4.4 Device Drivers

- 4.4.1 Block and Character Devices
- 4.4.2 Major and Minor Numbers

4.5 System Startup

4.6 System Shutdown

4.7 Redirection

- 4.7.1 Standard Input
- 4.7.2 Standard Output
- 4.7.3 Standard Error

4.8 Piping

4.9 Let Us Sum Up

4.10 Further Readings

4.11 Answer to Check Your Progress

4.12 Model Questions

4.1 LEARNING OBJECTIVES

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

- learn about Linux kernel
- describe the architecture of the Linux kernel
- describe the role of the kernel in the Linux system
- define device driver and its types
- explain how a device is represented in Linux system
- know about redirection and piping

4.2 INTRODUCTION

From the previous units, we have already learned the basic concepts along with its architecture and the basic features of the Linux operating system. We are also accustomed with the basic ideas about the various Linux Distributions. In the last unit, the concept of the linux installation process & requirements for the installation, how to partition the hard drive in linux, how to install the linux etc. were described. Besides these, we have also learned to install and configure different software or application in linux in the previous units.

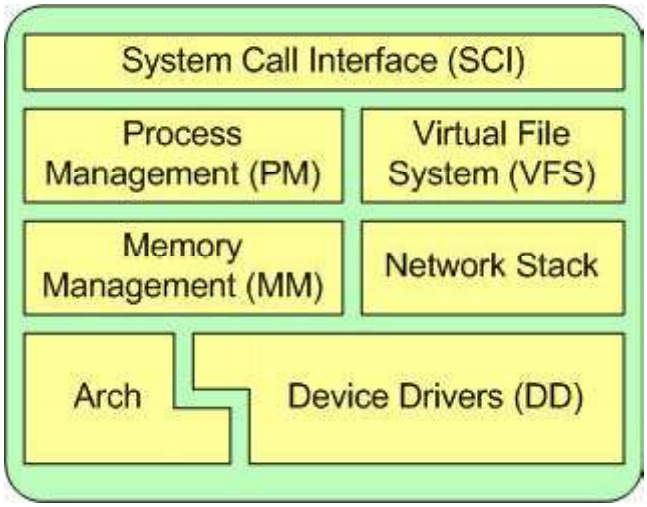
In this unit, we are going to discuss about the Linux Kernel, the role of the kernel in the linux system and the architecture of the Linux kernel. By going through this unit, we will be able to know about the concepts of device drivers in linux system in addition to booting and shutting down process of the system. The important features such as Redirection & Piping and how they work will also be learnt in this unit besides knowing about the standard input, standard output and the standard error in linux system.

4.3 LINUX KERNEL CONCEPTS

Kernel is a special code written in C language which directly interacts with hardware. It is at the core of the Linux system. Kernel runs simultaneously a number of processes. It acts as a manager for various resources like CPU, memory, process etc. Linux kernel is efficient in terms of both memory and CPU utilization. Kernels are of two types - monolithic kernel and micro kernel.

Linux kernel is a monolithic kernel because it accumulates all of the basic services. The linux kernel is layered into a number of distinct subsystems. The kernel is conceptually composed of six major subsystems: System Call Interface, the Process Management, the Memory Management, the Virtual File System, the Network Interface, and the Device Driver.

The major subsystems of the kernel are presented in the figure 4.1.



4.3.1 Major Functions of Linux Kernel

The major functions performed by kernel are described below.

- **Process Management:** The kernel handles the creation and destruction of the processes and their connection to the input and output systems. The inter process communication, i.e., the communication among different processes is also handled by the kernel. Process management can provide the information about what is happening with a process, file allocated to process, state of process (like running, waiting, stop), etc.
- **Memory Management:** The memory management is the most important function of kernel. Kernel assigns the address space to processes and applications, basically assigning virtual memory instead of physical memory, whereby later the actual address space in RAM is assigned and the conversion of physical address to virtual address is done by MMU (Memory Management Unit).
- **File System Management:** In Linux, almost everything is considered as a file. The Linux file system is a hierarchically data structured tree starting from root followed by other subdirectories which organizes the file system. A directory is a group of files. Two types of directories available in Linux system include: 1) root and 2) Sub directory. The root is the main

directory and sub directories can be created or renamed by the user under the root directory. The Linux file system supports a number of file types like ext2, ext3, device file, block file, networking file system etc. and each file system is stored in different disk partition.

- **System Call Interface:** The system call interface accomplishes function calls from user space into the kernel. It can be architecture dependent, even within the same processor family.
- **Device controlling:** This subsystem which maps to the different physical devices need to perform device control operations which are performed by code that is specific to the device being addressed. Those codes are called device drivers. For every peripheral connected to a system, there is a device driver embedded into the kernel.
- **Network Management:** This function is responsible for delivering data packets across program and network interfaces. The kernel handles the routing and address resolution issues in networking.

4.3.2 Architecture of Linux Kernel

The architecture of the Linux kernel is divided into two levels - User Space and Kernel Space. The architecture is depicted in the figure 4.2.

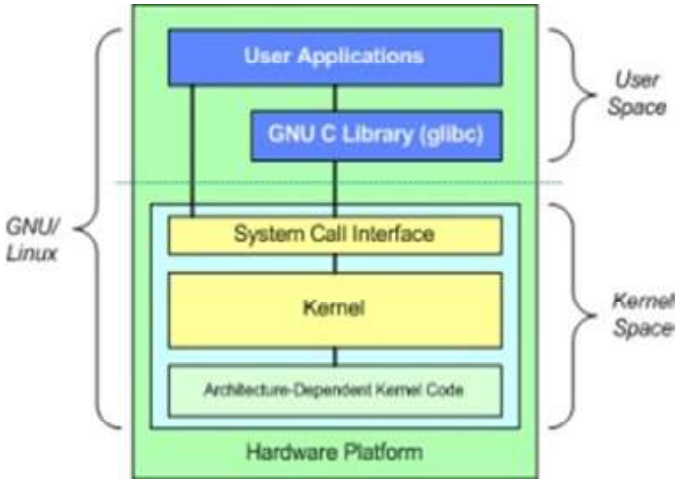


Fig. 4.2: Architecture of Linux Kernel

User Space: All the user programs and applications are executed in User Space. This Space cannot directly access the memory and hardware. It accesses those through kernel space. Hence, it needs to be switched user space application to kernel space and the switching mechanism is provided by the GNU C library.

Kernel Space: Kernel Space is for executing kernel programs. It can access full part of memory and directly interact with the hardware. The kernel space is further subdivided into three levels - System Call Interface, Kernel and Architecture-Dependent Code. System Call Interface sits in between the user space and kernel space and it executes the basic functions such as read and write. Kernel is architecture-independent code which is common to all hardware processors. The architecture-dependent code serves as the processor and platform specific code for the given architecture.

To summarise, the kernel space is the memory area that is reserved to the kernel while user space is the memory area reserved to a particular user process. The kernel space is accessed protected so that user applications cannot access it directly, while user space can be directly accessed from code running in kernel mode.

4.3.3 Features of Linux Kernel

As discussed in previous units, Linux is an open-source operating system. It supports a large number of networking protocols, including the typical TCP/IP. Linux kernel is a dynamic kernel that supports the addition and removal of software components on the fly by employing dynamically loadable kernel modules. And they can be inserted at booting time when they are required. Though much of Linux is independent of the architecture on which it runs, but there are some elements which are considered for normal operation and for efficiency reason. Low-level Scheduling of Processes, Inter-

process Communication, Process Synchronization and Context Switching are some other features of Linux kernel.

4.4 DEVICE DRIVERS

The software that handles or manages a hardware controller is known as a device driver. The Linux kernel device drivers are, essentially, a shared library of privileged, memory resident, low-level hardware handling routines. It is Linux's device driver that handles the peculiarities of the devices they are managing. The device drivers have a special role in the kernel of the Linux system. All the devices are represented by files. Nearly all system operations map to different physical devices and all device control operations are performed by some routines that are specific to a device. Those routines are referred to as device driver. The kernel calls the appropriate device driver, whenever a particular device is accessed and then the kernel passes some parameter for it to act properly. The device files are stored in /dev directory or in its sub-directories. The device files contain no data. The following Linux command lists device names connected to the system.

```
$ ls -l /dev
total 0
crw----- 1 root root 10, 55 Feb 3 18:11 acpi_thermal_rel
crw--- 1 root root 10, 235 Feb 3 18:11 autofs
drwxr-xr-x 2 root root 400 Feb 3 18:10 block
drwxr-xr-x 2 root root 80 Feb 3 18:10 bsg
crw--- 1 root root 10, 234 Feb 3 18:10 btrfs-control
drwxr-xr-x 3 root root 60 Feb 3 18:10 bus
lrwxrwxrwx 1 root root 3 Feb 3 18:11 cdrom -> sr0
lrwxrwxrwx 1 root root 3 Feb 3 18:11 cdrw -> sr0
drwxr-xr-x 2 root root 3900 Feb 3 18:11 char
crw--- 1 root root 5, 1 Feb 3 18:11 console
lrwxrwxrwx 1 root root 11 Feb 3 18:10 core -> /proc/kcore
drwxr-xr-x 2 root root 60 Feb 3 18:10 cpu
```

```
crw--- 1 root root 10, 59 Feb 3 18:11 cpu_dma_latency
crw--- 1 root root 10, 203 Feb 3 18:10 cuse
drwxr-xr-x 8 root root 160 Feb 3 18:10 disk
drwxr-xr-x 2 root root 80 Feb 3 18:10 dri
crw--- 1 root root 246, 0 Feb3 18:11 drm_dp_aux0
crw--- 1 root root 246, 1 Feb 3 18:11 drm_dp_aux1
crw--- 1 root root 246, 2 Feb 3 18:11 drm_dp_aux2
lrwxrwxrwx 1 root root 3 Feb 3 18:11 dvd -> sr0
lrwxrwxrwx 1 root root 3 Feb 3 18:11 dvdrw -> sr0
crw--- 1 root root 10, 61 Feb 3 18:11 ecryptfs
crw-rw--- 1 root video 29, 0 Feb 3 18:11 fb0
lrwxrwxrwx 1 root root 13 Feb 3 18:10 fd -> /proc/self/fd
crw-rw-rw- 1 root root 1, 7 Feb 3 18:11 full
```

4.4.1 Block and Character Devices

The Linux system classifies most of the hardware devices into two forms: block or character devices. These devices are indicated by the first character (either b or c) in the permission field. The character b indicates that it is a block device file and if it is c then it means that it is a character device file. A block device can store or transfer a block of data at a time, wherein a character device can display, process or store a character at a time.

The floppy drive, CD-ROM and the hard disk drive are the block special devices. All data are read from or written into these devices in blocks and use the buffer cache. The terminal, tape drive and printer are the character special devices. The read/write operations for character devices do not use the buffer cache and access the device directly.

4.4.2 Major and Minor Numbers

The fifth field in the file list representing the size for the other files, consists of a pair of two numbers, separated by a comma. These

numbers called the major and minor device numbers respectively. The major number represents the type of device and the minor number represents the special characteristics of the device. For example, tty0 and tty1 have the same major number (4) but different minor numbers (0 and 1) which is presented below.

crw-w-- 1 root tty 4, 0 Feb 3 18:11 tty0

crw-w-- 1 root tty 4, 1 Feb 3 18:11 tty1

4.5 SYSTEM START UP

To start-up the system, we have to power on the machine such that the power is supplied to the peripherals those are immediately required. When the kernel is loaded, these peripherals are also to be loaded. The start-up or booting process follows through a number of stages.

At first, the kernel is loaded into the memory. It is represented by the file /vmlinuz. BIOS (Basic Input/Output System) then performs some system integrity checks. It searches, loads, and executes the boot loader program from floppy, CD-ROM, or hard drive. In simple terms, BIOS loads and executes the MBR (Master Boot Record) which is located in the 1st sector of the bootable disk. The MBR then loads and executes the GRUB (Grand Unified Bootloader) that has the knowledge of the file system. The GRUB configuration file is /boot/grub/grub.conf. This file loads and executes kernel and then the kernel starts spawning further processes, among which the most important is init. The kernel executes init process by executing the program /sbin/init. Since, init is the first process to be executed by the kernel, its PID (process id) is 1. The init then in turn spawns further processes. The init controls a number of modes which is called the run level. The available run levels are-

- 0 - halt
- 1 - Single user mode
- 2 - Multiuser, without NFS
- 3 - Full multiuser mode
- 4 - unused

- 5 - X11
- 6 - reboot

If we look at the /etc/inittab file to decide the Linux run level, it may observe that init identifies the default init level from /etc/inittab and uses that to load all appropriate program. Depending on the default init level, the system will execute the programs from one of the following directories.

o Run level 0 - /etc/rc.d/rc0.d/

o Run level 1 - /etc/rc.d/rc1.d/

o Run level 2 - /etc/rc.d/rc2.d/

o Run level 3 - /etc/rc.d/rc3.d/

o Run level 4 - /etc/rc.d/rc4.d/

o Run level 5 - /etc/rc.d/rc5.d/

o Run level 6 - /etc/rc.d/rc6.d/

4.6 SYSTEM SHUTDOWN

The shutdown command is used to bring the system down in a secured way. The /sbin/shutdown command controls the shutdown sequence. This shutdown job is executed by signalling the init process, asking it to change the run level. The activities performed by shutdown are-

- Informs the users that the system is going down.
- Notifies all the running processes that the system is going to be shut down by sending the signals so that the processes can terminate normally.
- Unmounts all the secondary file system.

Different options used with shutdown command can be explained with following example.

Example 4.1: Shutdown examples

$ shutdown 8:00

Schedule the system to shut down at 8 A.M.

$ shutdown 20:00

Schedule the system to shut down at 8 P.M.

$ shutdown now

Bring down the system immediately.

$ shutdown -r now

Bring down the system immediately, and automatically reboot it.

$ shutdown -P now

Bring down the system immediately, and automatically power off the system.

4.7 REDIRECTION

Redirection is a special feature of Linux system through which one can direct the input or output to some other stream other than the standard ones. The redirection operators in Linux include >, < and >> where < means the input redirection, > specifies the output redirection operator and >> operator appends output to an existing file.

The input redirection operator (<) instructs the shell to redirect the input. It can be used to take input from a file. Suppose the thisfile.txt contains the following texts.

This is a text file.

It is used to test the cat and the wc commands.

The redirection and the piping can also be tested by using this file.

And this is the end of the content of the file.

The wc command will count and display the number of lines, words and characters in the file thisfile.txt.

$ wc < thisfile.txt

4 40 187

Here, 4 is the number of lines, 40 is the number of words and 187 is the number of characters.

The output of any command can be redirected by using the output redirection operator. For example, the command:

$ cat thisfile.txt > thatfile.txt

will write the contents of the file into another file thatfile.txt.

The input and output redirection operators can also be combined to

redirect both the input and output. For example,

$ wc < thisfile.txt > count_file.txt

command will save the number of lines, word and characters of the file ''thisfile.txt'' into another file ''count_file.txt''.

$ cat count_file.txt

4 40 187

When a command begins running, it usually expects that three files are already open: standard input, standard output, and standard error. A number, called a file descriptor, is associated with each of these files, as follows:

File descriptor 0 means Standard input

File descriptor 1 means Standard output

File descriptor 2 means Standard error (diagnostic) output

A child process normally inherits these files from its parent. All these three files are initially assigned to the peripherals like 0 to the keyboard, 1 and 2 to the display. The shell permits them to be redirected elsewhere before control is passed to a command.

When you enter a command and no file name is given then your keyboard is the standard input, denoted as stdin. When a command finishes, the results are displayed on your screen. This is called the standard output, denoted as stdout. By default, commands take input from the standard input and send the results to standard output. Standard Error, denoted as stderr, is where error messages go. By default, this refers to monitor screen.

4.7.1 Standard Input

The standard input is a character stream that comes from a particular source. A source may be a device or a file or a program or command. Some commands (like cat and wc) can operate on a character stream without knowing the source of the stream. If we write cat and wc command without any argument as follows-

cat

wc

Then these commands will receive character stream from the default source (i.e., from the keyboard). This stream is called standard input. The sources from which the standard input can come from are-

- the keyboard, which is the default source.
- a file (given by using the redirection operators >, <, >>)
- a program or command (given by using pipe)

For example,

$ wc
Hello, I am wc command.
can count the number of lines, words and characters.
<Ctrl d>
2 15 79
$ wc < thisfile.txt
4 40 187

4.7.2 Standard Output

Standard output writes the output generated by a program or command in the form of a character stream. When the standard output is not redirected, it will send the output to the terminal, which is the default destination.

For example, if you write the cat and wc command as follows-

cat
wc

Then these commands will send the output as a character stream to the terminal. This stream is called the standard output. Here, the shell sets up a connection between the command and the terminal to display the output on the monitor screen. Again, the standard output can be redirected.

For example,

$ wc thisfile.txt > thatfile.txt

This command redirects the number of lines, number of words and the number of characters in the file thisfile.txt to

the file thatfile.txt.

The standard output can also serve as input to another program. Thus, the standard output has three destinations - the terminal (which is default destination), a file (redirected by using redirection operator) and another command or program (given by suing the concept of piping).

4.7.3 Standard Error

The standard error writes error messages generated by a program or command that fails at some point in its execution. It writes the error message to the terminal. Like standard output, the default destination of standard error is the terminal display. The output redirection operator can also be used to redirect the error to a file. But it needs to prefix the file descriptor. A file descriptor is a number assigned to a standard file that identifies the standard file. The file descriptors 0,1 and 2 represent the standard input, standard output and standard error respectively.

For example,

$ cat testfile.txt > newfile.txt

cat: testfile: No such file or directory

$ cat testfile 2> newfile

Here, standard error has been redirected by using file descriptor.

$ cat newfile

cat: testfile: No such file or directory

4.8 PIPING

A pipe is a form of redirection (transfer of standard output to some other destination) that is used in Linux. Pipes are used to redirect a standard input or standard output from one command or program to another for further processing. The shell enables the standard output of one command to be connected as a standard input to another. This is called piping.

Piping is done by using a special operator "|" called pipe that acts

as the connector of two commands. Here, the output of one command acts as input to the next command and so on. It can also be visualized as a temporary connection between two or more commands/ programs/ processes. Consider, there are four text files which are listed as:

$ ls

file1.txt file2.txt file3.txt file4.txt

$ ls | wc

4 4 40

Here, the ls command passes its output directly to the input of wc then the wc command gives the output as above.

$ ls file1.txt | wc

1 1 10

$ cat file1.txt | wc

1 5 19

Here, the cat command passes its output (i.e., the content of the file file1.txt) to the input of wc command and wc command gives the final output which is displayed on the screen.

Pipes are unidirectional i.e., data flows from left to right through the pipeline. The syntax is as follows:

command_1 | command_2 | command_3 | .... | command_N

For example, to list all files and directories and give it as input to more command, the following command with pipe displays the output as given below.

$ ls -l | more

Output is (Fig. 4.3):

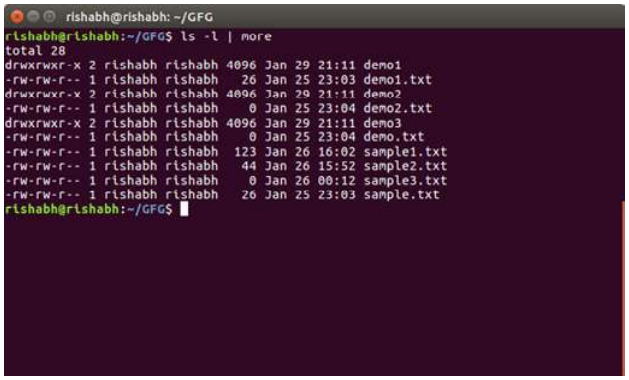


Fig. 4.3: Output of the command "ls -l | more"



CHECK YOUR PROGRESS

Choose the correct option and fill up the blanks:

Q 1: Redirection is a process of switching of the standard stream of data.

a) True b) False

Q 2: How many files are used for representing different standard streams?

a) 1 b) 2 c) 4 d) 3

Q 3: Which stream is connected to the display?

a) standard input b) standard output

c) standard error d) None of these

Q 4: The command wc < sample.txt will count data from the file sample.txt.

a) True b) False

Q 5: Which of the following symbol(s) can be used to redirect the output to a file or another program?

a) | b) > c) >> d) |, > and >>

Q 6: Device Drivers reside in which environment?

a) Kernel space b) Device space

c) Application space d) User space

Q 7: Which command is used to display the operating system name

a) os b) unix c) kernel d) uname

Q 8: A ___ is a special program that can control a specific type of hardware. (Pick the right option)

a) Shell b) Device driver c) Complier d) Database

Q 9: _____ directory has system configuration files, network configuration files, application configuration files and other installed software. (Pick the right option)

a) /var b) /mnt c) /opt d) /etc

Q 10: In Linux, devices are placed in _____ directory. (Fill up the blank)



4.9 LET US SUM UP

- Kernel is the core component of Linux system that interact with the hardware.
- Each device is considered a file.
- A device driver is a routine of codes specific to a particular device that performs the control operation for that device.
- In Linux, the system start-up process goes through a series of stages. At first, the kernel is loaded into the memory. BIOS (Basic Input/Output System) then performs some system integrity checks. It searches, loads, and executes the boot loader program from floppy, CD-ROM, or hard drive.
- Shutdown command is used with several options for shutting down a system.
- Redirection is used to redirect the output to some other stream other than the standard inputs and outputs.
- The standard inputs, standard outputs and the standard errors are the character streams.
- Piping is the technique of redirecting a standard input or output from one command to another. Piping is done by using a special operator "|" called pipe that acts as the connector of two commands.



4.10 FURTHER READINGS

1) Das, S. (2000).UNIX: Concepts and Applications. Tata McGraw-Hill.
2) Negus, C. (2012). Linux Bible. (Vol. 772). John Wiley & Sons.



4.11 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: Q.1) a Q.2) d Q.3) b Q.4) a
Q.5) c Q.6) a Q.7) d Q.8) b Q.9) d
Q.10) /dev



4.12 MODEL QUESTIONS

Q 1: Define Kernel? Describe the functions and the architecture of the Linux kernel.

Q 2: What do you mean by device driver? How can you identify whether a device file is block or character device?

Q 3: Describe the phases performed by the start-up process of Linux system.

Q 4: Describe the various options used with the shutdown command in Linux.

Q 5: What are redirection and piping? Explain the functionality of these two techniques.

Q 6: What do you mean by file descriptor? State the difference between standard input, standard output and the standard error.

*** ***** ***