GCA (S5) 01
Exam. Code: LSA

Linux System Administration

SEMESTER V

BACHELOR OF COMPUTER APPLICATION

Block 2



KRISHNA KANTA HANDIQUI STATE OPEN UNIVERSITY

Subject Experts

Prof. Anjana Kakati Mahanta, Gauhati University

Prof. (Retd.) Pranhari Talukdar, Gauhati University

Dr. Jyotiprokash Goswami, Associate Professor, Assam Engineering College

Course Co-ordinators

Dr. Tapashi Kashyap Das, Assistant Professor, KKHSOU

Ms. Sruti Sruba Bharali, Assistant Professor, KKHSOU

SLM Preparation Team

UNITS	CONTRIBUTORS
9	Ms. Sruti Sruba Bharali, KKHSOU
10	Ms. Sangeeta Borkakoty, USTM
11	Mr. Bhupesh Talukdar, Dimoria College
12	Mr. Biswajit Das, Cotton University
13 & 15	Mr. Manabendra Nath, NEIRM
14	Mr. Dwipen Laskar, Gauhati University

Editorial Team

Content : Mr. Gautam Chakrabarty, NERIM

Language : Prof. (Retd.) Robin Goswami, Cotton College

Structure, Format & Graphics: Ms. Sruti Sruba Bharali, KKHSOU

June, 2019 ISBN: 978-93-89123-74-6

This Self Learning Material (SLM) of the Krishna Kanta Handiqui State University is made available under a Creative Commons Attribution-Non Commercial-ShareAlike4.0 License (International): http.//creativecommons.org/licenses/by-nc-sa/4.0.

Printed and published by Registrar on behalf of the Krishna Kanta Handiqui State Open University.

Headquarters : Patgaon, Rani Gate, Guwahati-781 017

City Office : Housefed Complex, Dispur, Guwahati-781 006; Web: www.kkhsou.in

The university acknowledges with thanks the financial support provided by the **Distance Education Bureau, UGC** for the preparation of this study material.

BACHELOR OF COMPUTER APPLICATION

Linux System Administration

Block-2

CONTENTS

		Pages
UNIT 9:	**LINUX Shells** Study of Different Linux Shells (sh, bash, csh, zsh), Environment Variables, Shell Script Basics (Examples of Some Simple Shell Programming)	**137-153**
UNIT 10:	**Process** Basic Commands for Starting and Stopping Processes, Basic Process Attributes and their Role in Access Control. Examining the list of Running Processes on the System and Understand the Data Presented There. Background Process, Job Control, Crontab File Format, Backup and Restore Procedure	**154-173**
UNIT 11:	**Printing Jobs** Submit a Print Job, Check the Status of a Print Job, Cancel a Print Job, Configuring the Print Queue, Selecting the Print Driver, Editing the Printer Configuration	**174-186**
UNIT 12:	**LINUX Users** Understanding the “root” Account, Becoming a Superuser (su), A Limited su (sudo) Managing User Accounts– Adding a New User, Modifying and Removing User Accounts, Changing Password, System Monitoring and Logging, Monitoring Memory Usage, Disk Space Usage and I/O Activity	**187-198**
UNIT 13:	**Networking in LINUX** The Rules Governing IP Address Classes and Netmasks, Network Address, Netmask and Gateway Configuring Interface with ifconfig, ping, netstat, traceroute, telnet, Understanding the Significance of the /etc/services file and Well Known Port Numbers	**199-218**
UNIT 14:	**Network Protocols** Basics of Configuring NFS, NIS, DNS, FTP, Squid Proxy, DHCP Server, ip Tables and Firewall	**219-238**
UNIT 15:	**Basic Network Security Issues** Introduction to Basic Network Security Issues, Packet Snifflers, DOS Attacks, Linux Kernel Firewalling, Virtual Private Networks	**239-257**

BLOCK INTRODUCTION

This is the second block of the course ***Linux System Administration***. After completing this block, learners will be able to write shell programming scripts, understand how process work and how printing jobs are managed in Linux systems. Learners will also be able to manage Linux users and perform networking in Linux using knowlegde of different networkig protocols after reading this block. Learners will also be able to protect your Linux system using different security aspects of any Linux OS.

This block comprises the following seven units:

Unit 9 introduces us to basics of shell scripting. Basics of shell programming along with introduction to environent variables are mentioned in this unit.

Unit 10 deals with process in Linux operating system. The basic process attributes, list of running processes, job control, backup and restore are some of the topics discussed in this unit.

Unit 11 describes printing jobs in Linux. The procedure to submit and cancel a printing jobs is discussed in this unit along with the procedure to configure a printing queue.

Unit 12 introduces us to the Linux users. Creating users, modifying and removing users, becoming superusers, changing password are some of the topics dicussed in this unit.

Unit 13 deals with networking in Linux. Topics like rules for IP address and netmasks, defining gateways, understanding etc/service files are covered in this unit.

Unit 14 deals with network protocols in Linux. Protocols like NIS, NFS, DNS, FTP etc along with squid proxy, DHCP server etc are described in detail in this unit.

Unit 15 gives us an introduction to the network security aspects of Linux. Topics like denial of service attacks, packet sniffers, firewalling, virtual private networks are described in this unit.

Each unit of this block includes some along-side boxes to help you know some of the difficult, unseen terms. Some “EXERCISES” have been included to help you apply your own thoughts. You may find some boxes marked with: “LET US KNOW”. These boxes will provide you with some additional interesting and relevant information. Again, you will get “CHECK YOUR PROGRESS” questions. These have been designed for you to self-check your progress of study. It will be helpful for you if you solve the problems put in these boxes immediately after you go through the sections of the units and then match your answers with “ANSWERS TO CHECK YOUR PROGRESS” given at the end of each unit.

UNIT 9: LINUX SHELLS

UNIT STRUCTURE

9.1 Learning Objectives

9.2 Introduction

9.3 Linux Shell

9.4 Different Types of Shell (sh, bash, csh, zsh)

9.5 Environmental Variables

9.6 Shell Programming Basics

9.7 Let Us Sum Up

9.8 Further Reading

9.9 Answers to Check Your Progress

9.10 Model Questions

9.1 LEARNING OBJECTIVES

After going through this unit you will be able to:

- explain Linux shell
- describe different types of shell
- define environment variables
- write and run shell script programs.

9.2 INTRODUCTION

In the previous units in block 1 you have been introduced to many Linux commands. We have also learned different types of file handling commands and filter commands. Commands for granting and removing group and user permissions were also covered in detail in the last unit.

In the present unit you will be introduced to the concepts of Linux shell. An introduction to different types of shell and environment variables is covered in this unit. Shell programming basics have also been introduced in this unit. Starting from the basic program to using functions and arrays in shell has been included in this unit.

In the next unit you will be made familiar with the basics of process in Linux. Commands for starting and killing a process along with commands for managing jobs in Linux will be covered in the next unit.

9.3 LINUX SHELL

A **Shell** provides an interface to the Linux system. It provides an environment in which we can run our commands, programs, and shell scripts. Shell is not part of system kernel, but it uses the system kernel to execute programs, create files etc.

Shell can be accessed by the user using a command line interface. A special program known as the **Terminal** is used to provide and execute commands in shell. The terminal is a program that opens a window and lets you interact with the shell. The shell gets started when the user logs in or starts the terminal. Shell can also be defined as a command language interpreter that executes commands read from the standard input device such as keyboard or from a file. The result can also be displayed on the terminal to the user.

Apart from the terminal shell can also be accessed using other ways. **Connecting via secure shell (SSH)** helps the users to log in into remote server or workstation. Some Linux systems also provide a text-based login system called the **Console.**

9.4 DIFFERENT TYPES OF SHELL

There are different types of a shell. Each type of shell has its own set of recognized commands and functions. The syntax of writing commands may vary among these different types of shells. Each shell will perform the same job even though different commands and built-in functions may be used.

There are two major types of shells– **Bourne shell** and **C shell**. They are defined below.

- **Bourne shell (sh):** The Bourne shell was developed by Stephen Bourne at Bell Labs and released in 1979 in the Version 7 Unix. It was a replacement for the Thompson shell, whose executable file

had the same name– sh. Most Unix-like systems continue to have / bin/sh –which is the Bourne shell. The $ character is the default prompt in the Bourne-type shell.

The Bourne Shell has the following subcategories–

- **Korn shell (ksh):** The Korn shell is written by David Korn based on the original Bourne Shell source code. It is a middle road between the Bourne shell and the C shell. Its syntax was chiefly drawn from the Bourne shell, while its job control features resembled those of the C shell. The Korn Shell also was the base for the POSIX Shell standard specifications etc.
- **Bourne Again shell (bash):** Bourne Again shell was developed in 1989 for the GNU project and incorporates features from the Bourne shell, csh, and ksh. It is meant to be POSIX-compliant. It is the most widely used shell in Linux systems. It is used as default login shell in Linux systems and in macOS. It can also be installed on Windows OS.

- **C shell (csh):** The C shell is a UNIX shell created by Bill Joy. It is a command processor typically run in a text window, allowing the user to type commands. The main design objectives for the C shell were that it should look more like the C programming language and that it should be better for interactive use. The % character is the default prompt in the C-type shell.

On many systems, such as Mac OS X and Red Hat Linux, csh is actually tcsh, an improved version of csh. On Debian and some derivatives (including Ubuntu), there are two different packages: csh and tcsh. The former is based on the original BSD version of csh and the latter is the improved tcsh. tcsh added filename and command completion and command line editing concepts borrowed from the Tenex system, which is the source of the “t”.

9.5 ENVIRONMENT VARIABLES

Environment variables are an important part of the LINUX system. These variables are set by the system, or by users or by the shell, or any

program that loads another program. Environment variables are dynamic values which affect the processes or programs on a computer. These variables exist in every operating system though the types of environment variables may vary depending on the operating system. Environment variables can be created, edited, saved, and deleted and they give information about the system behavior.

Let us now look at the definition of some of the environment variables.

Environment Variable	Description
PATH	Indicates the search path for commands. It is a colon-separated list of directories in which the shell looks for commands.
HOME	Indicates the home directory of the current user
USER	Displays the user name
SHELL	Displays the path name of the home directory
UID	Expands to the numeric user ID of the current user, initialized at the shell startup
TERM	Refers to the display type
EDITOR	Gives the name of the preferred text editor
TZ	Refers to Time zone



CHECK YOUR PROGRESS

Q.1: Define shell.

.....

.....

Q.2: What are the two major types of shell?

.....

Q.3: What are environment variables?

.....

.....

9.6 SHELL PROGRAMMING BASICS

In this section we will now try to understand how to write shell script programs. The first program is to display a simple "Hello World" as the output. The program is given in program 9.1.

Program 9.1: Program to display "Hello World"

```
# ! /bin/bash
# My first script
echo Hello World
echo 'Hello       World'
echo "Hello                    World"
```

Run/Execution:

```
chmod 755 hello.sh
./hello.sh
```

Output:

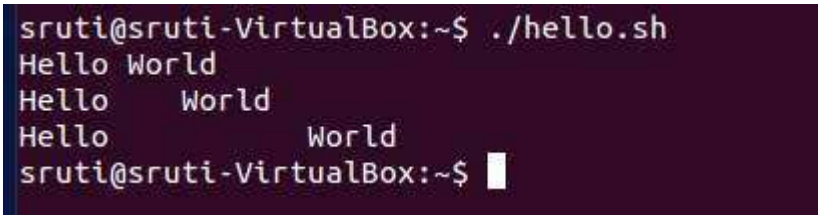


Figure 9.1: Output for Program 9.1

The first **line starting with # represents that it is a comment** in the shell script. The echo command in shell is equivalent to a printf() function in C programming. The "hello.sh" shell script contains three lines to display the words "Hello World".

The command **chmod 755 *filename*** is used to make the shell script file "***filename"*** executable. Here, the command **chmod 755 hello.sh** is used to make the shell script file "hello.sh" executable. Once the file is executable then we can run file by **"./"** before the name of the file, for example **./hello.sh** in this case. The output for program 9.1 is displayed in the figure 9.1. The output shows three lines of "Hello World". It can be seen from the output that the space between the words Hello and World as given is the input is displayed in the output display.

Let us look at program 9.2 in which a variable will be created and the value of the variable will be displayed at the output.

Program 9.2: Program to display value of a variable

```
#!/bin/bash
a=1
echo $a
```

Output:

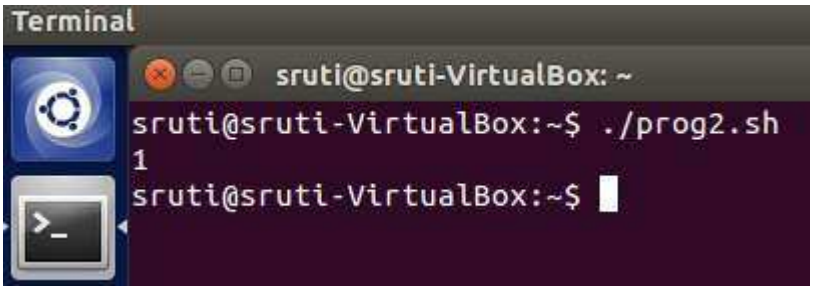


Figure 9.2: Output for Program 9.2

The first line of the program is a comment. The second line declares a variable “a” which is assigned a value “1”. The third line of the program uses the echo command to display the value of the variable “a” on the terminal screen. The output for program 9.2 shows the value of the variable “a” on the terminal screen.

Program 9.3: Program showing use of “if-elif-else” condition using shell script

```
#!/bin/bash
#!/bin/sh
if [ $1 -gt 0 ]; then
  echo "$1 is positive"
elif [ $1 -lt 0 ] then
  echo "$1 is negative"
elif [ $1 -eq 0 ] then
  echo "$1 is zero"
else
 echo "Opps! $1 is not number"
fi
```

Run/Execution:

./ prog3.sh 5

Output:

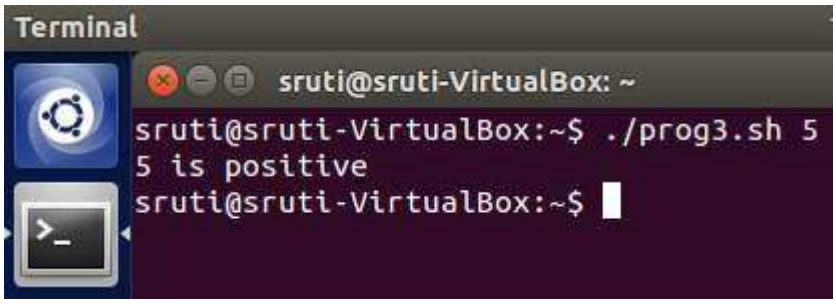


Figure 9.3: Output for Program 9.3

The first two lines of the program are comment lines. The program starts from the third line using if statement where the user input represented by "$1" will be compared to see if it is greater than (gt) zero. In case this condition is true, hence then the remaining statements of the if-condition are executed. If the case was that this condition is false, then the next statement will be executed which is the else-if statement. The user input would then be compared to see if it is lesser than (lt) zero and if this is true than the remaining statements for this condition will be executed. In the case that this condition is also false, then the next statement will be executed. The user input would then be now compared to see if it is equal to (eq) zero and if this is true then the remaining statements for this condition will be executed. If all the conditions are false then it enters the else condition and the statements in else condition part is executed. The if-elif-else statements are closed using the keyword **"fi",** which is the reverse of ''if'' keyword.

The output for program 9.3 is shown in figure 9.3. As explained earlier, the program 9.3 requires a number as user input. In this case, a number "5" is given as the input. Using the if- elif- else statements the input number is compared with zero to check if the number is positive, negative or zero. Since the input given is "5", hence the output is shown to be positive.

Let us now look at an example where **case** statements are used in shell scripts.

Program 9.4: Program to show use of "case" statements using shell script

```
#!/bin/bash
case $1 in
   "car") echo "For $1 Rs.20 per k/m";;
   "van") echo "For $1 Rs.10 per k/m";;
   "jeep") echo "For $1 Rs.5 per k/m";;
   *) echo "Sorry, cannot get a $1l for you";;
esac
```

Run/Execution:

```
chmod +x prog4.sh
./ prog4.sh van
```

Output:

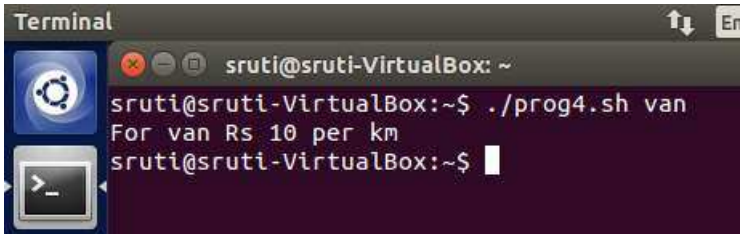


Figure 9.4: Output for Program 9.4

The "case" keyword signifies the starting of case block and 'esac" keyword signifies the ending of case block. The different cases are represented within " " quotation marks followed by ")" and the default case is represented by "*)".

Another way of executing a shell script is to make it executable by adding the executable part to the shell script. This is done by writing the **chmod** command and then the plus sign "**+**" along with "**x**" followed by the filename. For example to make the shell script prog4.sh executable, we will simply have to write **chmod +x prog4.sh**.

The output for program 9.4 is shown in figure 9.4. The program 9.4 requires a string as user input to select the variety of the motor vehicle used. In this case, the string "van" is given as the input. Here, three cases are presented in the program to represent "van, "car" and "jeep" along with

a default option represented by "*)". Since the input given is "van", hence the output shows the per km rate for "van".

Let us now look at an example where **for loop** statements are used in shell scripts.

Program 9.5: Program showing use of for loop in shell script

```
# ! /bin/bash
for i in 1 2 3 45
do
  echo "Welcome $i times"
done
```

Output:

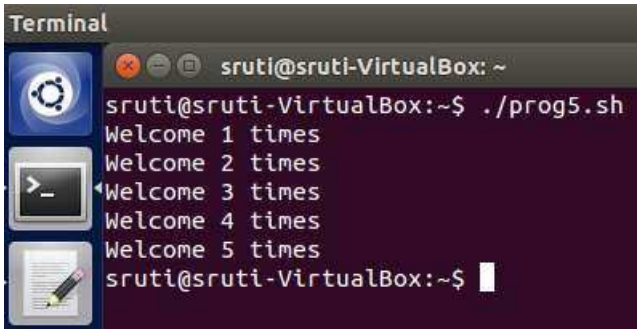


Figure 9.5: Output for Program 9.5

The first line of the program is a comment. The second line specifies the condition in for loop using the **for** command where the variable i goes from 1 to 5. The third line with command **do** marks the beginning of for loop and the fifth line with command **done** marks the end of the for loop. The fourth line in between displays the value of variable "i" each time the for loop is executed. The output for program 9.5 is shown in figure 9.5.

Let us now look at an example where **while loop** statements are used in shell scripts.

Program 9.6: Program showing use of while loop in shell script

```
# ! /bin/bash
a=0
while [ $a -lt 5 ]
do
```

```
    echo $a
    a=`expr $a + 1`
done
```

Output:

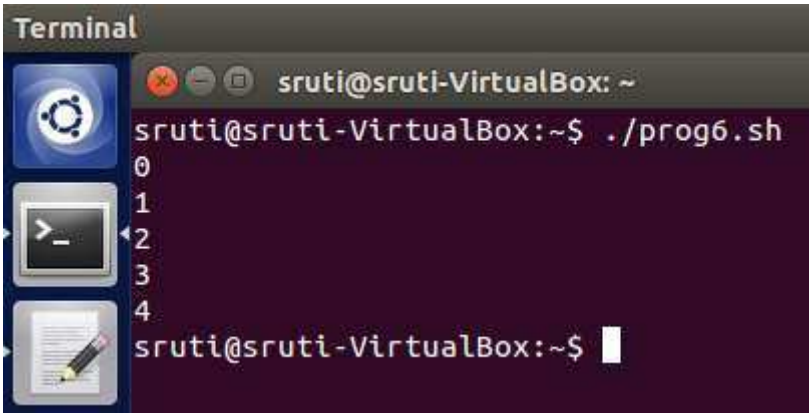


Figure 9.6: Output for Program 9.6

The first line of the program is a comment. In the second line of the variable “a” is initialized to “0”. The condition for the while loop is given in the third line using the **while** command. The **do** marks the start of the while loop statements and **done** marks the end of the while loop. The first statement within the while loop prints the value of the variable on the terminal and the second statement increases the value of the variable by 1. In this example, the condition is given for comparing if the variable “a” is less than “5”. While the value of the variable is less than 5 the while loop statements gets executed. In the event of the value of the variable not less than 5 the while loop is exited. The output for program 9.6 is shown in figure 9.6.

Functions in Shell Script: Like in other programming languages, functions can also be used in shell script. The syntax for declaring a function is given below.

```
function_name () {
  // list of commands
}
```

Here, function_name is the name of the function.

Let us now look at examples where **functions** are used in shell scripts.

Program 9.7: Program to show how to pass parameters to function

```
#!/bin/sh
Hello ()
{
   echo "Hello $1 $2"
}
# Invoke your function
Hello Dear Learner
```

Output:

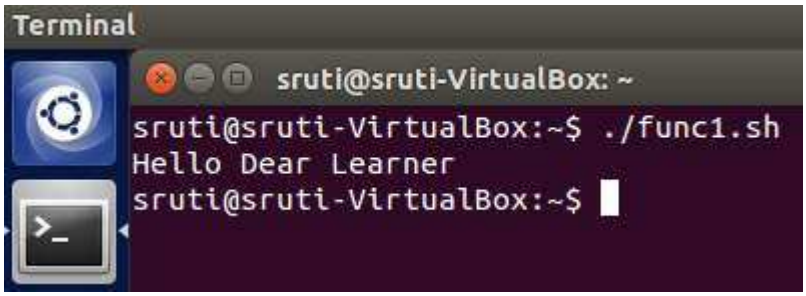


Figure 9.7: Output for Program 9.7

The first line of the program is a comment. In the second line the function "Hello()" is declared. The next line contains the definition of the Hello() function where we want to display the values of two variables using echo command. The fifth line of the program is again a comment. In the sixth line we invoke the function by giving the name of the function followed by any two parameters. In this example, the name of the function "Hello" is followed by two parameters "Dear" and "Learners". The output for program 9.7 is shown in figure 9.7.

Program 9.8: Program to show returning of values from functions

```
#!/bin/sh
Hello () {
   echo "Hello $1 $2"
   return 10
}
```

```
# Invoke your function
Hello Dear Learner
# Capture value returned by last command
ret=$?
echo "Return value is $ret"
```

Output:

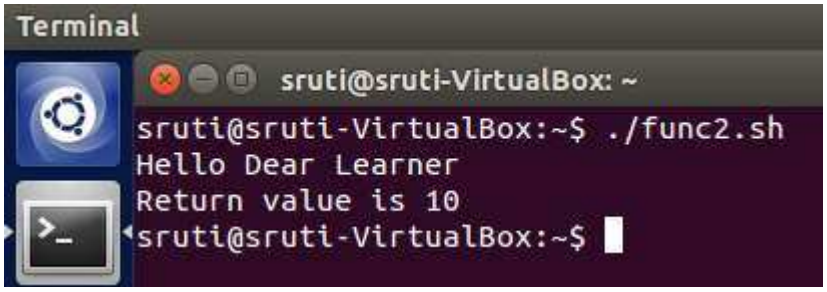


Figure 9.8: Output for Program 9.8

The first three lines of the shell script is same as the script before. In the fourth line we return a value of "10". The sixth line of the program is again a comment. In the seventh line we invoke the function by giving the name of the function followed by any two parameters as done in the example before. The eight line of the program is again a comment. In the ninth line we capture the value returned by the function Hello() in the variable "ret". The last line of the script displays the value of the variable "ret" using echo command. The output for program 9.8 is shown in figure 9.8.

Program 9.9: Program using nesting functions

```
#!/bin/sh
# Calling one function from another
number_one () {
   echo "First function"
   number_two
}
number_two () {
   echo "Second function"
}
```

```
# Calling function one

number_one
```

Output:

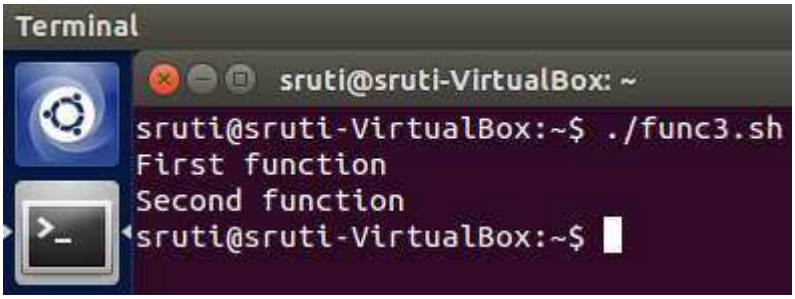


Figure 9.9: Output for Program 9.9

In this example, there are two functions - function “number_one()” and function “number_two()”. The function “number_one()” calls the function “number_two()” within its definition. In the last line of the function “number_one()” is invoked. Since, function “number_one()” calls the function “number_two” hence the output displays the echo statements for both the functions in the output. The output for program 9.9 is shown in figure 9.9.

Arrays in Shell Script: Like functions, arrays can also be used in shell script. The syntax for declaring an array is given below.

```
array_name[index]=value
```

Here **array_name** is the name of the array, **index** is the index of the item in the array that you want to set, and **value** is the value you want to set for that item. If you are using the bash shell, the syntax of array initialization is given below.

```
array_name = (value1 ... valueN)
```

After the initialization of array variables, we can access these arrays. The syntax for accessing a particular index of an array variable is given below.

```
${array_name[index]}
```

Here, **array_name** is the name of the array, and **index** is the index of the value to be accessed. Let us look at a shell script showing how to use arrays in shell script in program 9.10.

Program 9.10: Program showing how to use arrays in shell script

```
#!/bin/bash
array1[0]=one
array1[1]=1
echo ${array1[0]}
echo ${array1[1]}

array2=( one two three )
echo ${array2[0]}
echo ${array2[2]}
```

Output:

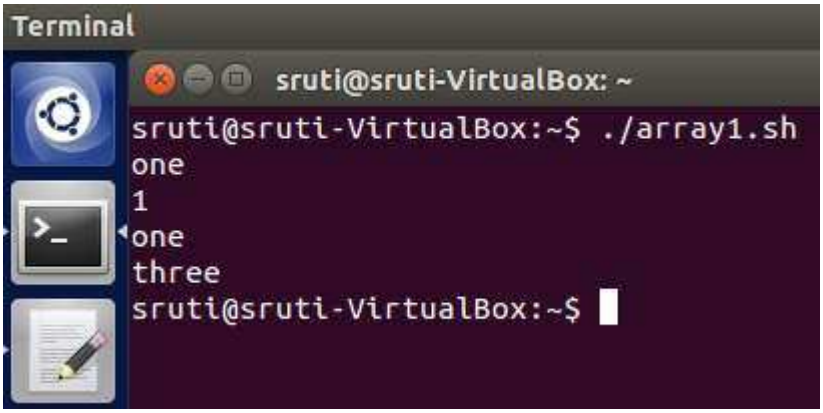


Figure 9.10: Output for Program 9.10

The first line of the shell script is a comment. In the second and third line of this script, the array “array1” is initialized with two values of “one” and “1”. In the fourth and fifth line, the first and second elements of the array “array1” are accessed and then displayed using the echo command. In the sixth line of the shell script another array “array2” is initialized with three values “one”, “two” and “three”. In the last two lines the array “array2” is accessed and the values of first and third elements in index “0” and “2” are displayed using echo command. The output for program 9.10 is shown in figure 9.10.



CHECK YOUR PROGRESS

Q.4: What is the use of **chmod 755** command?

.....

.....

Q.5: Give the syntax for declaring functions in shell script.

.....

.....

.....

.....

Q.6: Give the syntax for declaring arrays in shell script.

.....

.....

.....

.....



9.7 LET US SUM UP

- A shell provides an interface to the Linux system.
- Shell provides an environment in which we can run our commands, programs, and shell scripts.
- Shell is not part of system kernel, but it uses the system kernel to execute programs, create files etc.
- There are two major types of shells– Bourne shell and C shell.
- The Bourne shell was developed by Stephen Bourne at Bell Labs and released in 1979 in the Version 7 Unix.
- The C shell is a UNIX shell created by Bill Joy. The main design objectives for the C shell were that it should look more like the C programming language and that it should be better for interactive use.
- Environment variables are dynamic values which affect the processes or programs on a computer.

- The command **chmod 755 *filename*** is used to make the shell script file "***filename"*** executable.
- The syntax for declaring a function is:

```
function_name () {
  // list of commands
          }
```

  where, function_name is the name of the function.
- The syntax for declaring an array is:

```
array_name[index]=value
```

  where **array_name** is the name of the array, **index** is the index of the item in the array that you want to set, and **value** is the value you want to set for that item.



9.8 FURTHER READING

1) Kanetkar, Y. P. (2003). *UNIX Shell Programming*. BPB Publications.
2) http://www.freeos.com/guides/lsst/
3) https://www.tutorialspoint.com/unix/unix-what-is-shell.htm



9.9 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: A shell provides an interface to the Linux system. It provides an environment in which we can run our commands, programs, and shell scripts.

Ans. to Q. No. 2: The two major types of shells are– Bourne shell and C shell.

Ans. to Q. No. 3: Environment variables are dynamic values which affect the processes or programs on a computer.

Ans. to Q. No 4: The command **chmod 755 *filename*** is used to make the shell script file "***filename"*** executable.

Ans. to Q. No. 5: The syntax for declaring a function is:

```
function_name () {
      // list of commands
            }
```

where, function_name is the name of the function.

Ans. to Q. No. 6: The syntax for declaring an array is:

```
array_name[index]=value
```

where **array_name** is the name of the array, **index** is the index of the item in the array that you want to set, and **value** is the value you want to set for that item.



9.10 MODEL QUESTIONS

Q.1: What are environment variables?

Q.2: How is shell related to kernel in LINUX systems?

Q.3: How are functions declared in shell script? Explain using example.

Q.4: Describe any four environment variables.

Q.5: Differentiate between Bourne shell and C-shell.

Q.6: Describe the different types of shell.

Q.7: How are arrays declared in shell script? Explain using example.

Q.8: Write a program using shell script to display the first 10 Fibonacci numbers.

Q.9: Write a program using shell script to display the even natural numbers from 10 to 50.

Q.10: Write a program using shell script to find the factorial of a number.

*** ***** ***

UNIT 10: PROCESS

UNIT STRUCTURE

10.1 Learning Objectives

10.2 Introduction

10.3 Process

- 10.3.1 Starting a Process
- 10.3.2 Killing a Process

10.4 Basic Process Attributes and their Role in Access Control

- 10.4.1 Process Attributes
- 10.4.2 The Pid and Parentage
- 10.4.3 Credentials

10.5 Examining the List of Running Processes

10.6 Background Process

10.7 Job Control

- 10.7.1 Unique Identifiers for a Job
- 10.7.2 Managing Jobs in Linux

10.8 Crontab File Format

- 10.8.1 The Crontab File Fields

10.9 Backup and Restore Procedure

- 10.9.1 Using tar Command
- 10.9.2 Using cpio Command

10.10 Let Us Sum Up

10.11 Further Reading

10.12 Answers to Check Your Progress

10.13 Model Questions

10.1 LEARNING OBJECTIVES

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

- describe processes and their types
- start and stop processes
- describe process attributes and what role they play in access control
- explain controlling jobs in Linux

- schedule tasks using crontab file
- backup and restore files and directories.

10.2 INTRODUCTION

In the previous unit, we have learnt what is shell and what are the different types of shell. We have also learnt how to write and run shell scripts. In this unit, we will learn what is a process, and how to start and kill a process. We will also learn about background process and about job control and crontab file format. In the next unit, we will learn about managing printing jobs.

10.3 PROCESS

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

A program/command when executed, a special instance is provided by the system to the process. This instance consists of all the services/ resources that may be utilized by the process under execution.

- Whenever a command is issued in Linux, it creates/starts a new process. For example, pwd when issued which is used to list the current directory location the user is in, a process starts.
- Linux keeps account of the processes through a five digit ID number. This number is call process id or pid. Each process in the system has a unique pid.
- Used up pid's can be used in again for a newer process since all the possible combinations are used.
- At any point of time, no two processes with the same pid exist in the system because it is the pid that Linux uses to track each process.

There are fundamentally two types of processes in Linux:

- **Foreground processes** (also referred to as interactive processes)– these are initialized and controlled through a terminal session. In other words, there has to be a user connected to the system to start such

processes. They haven't started automatically as part of the system functions/services.

- **Background processes** (also referred to as non-interactive/automatic processes)– are processes not connected to a terminal.They run in the background without keyboard input and wait till keyboard input is required. Thus, other processes can be done in parallel with the process running in background since they do not have to wait for the previous process to be completed.

Any command that you give to your Linux machine starts a new process. Having multiple processes for the same program is possible. Generally, an application process lifecycle has three main states: *start, run,* and *stop*.

10.3.1 Starting a Process

The easiest way to start a process is to type its name at the command line and press Enter.

For example,

```
$ myprogram.sh ↵
```

A process may not continue to run when you log out or close your terminal. This special case can be avoided by preceding the command you want to run with the **nohup** command. Also, appending an ampersand (&) will send the process to the background and allow you to continue using the terminal.

For example, suppose you want to run myprogram.sh.

```
$ nohup myprogram.sh &↵
```

10.3.2 Killing a Process

The first step in killing an unresponsive process is locating it. There ***top*** command can be used to locate a process.

With ***top***, you get a full listing of currently running process.

```
$ top ↵
```

From this output, you can find the instances of the irresponsive process along with their PIDs (Process ID).

Now we come to the task of killing the process. We have two pieces of information that will help us kill the errant process:

- Process name
- Process ID

Which you use will determine the command used for termination.

There are two commands used to kill a process:

- kill – Kill a process by ID
- killall – Kill a process by name

There are also different signals that can be sent to both kill commands. What signal you send will be determined by what results you want from the kill command. For instance, you can send the HUP (hang up) signal to the kill command, which will effectively restart the process. You can get a list of all the signals that can be sent to the kill command by issuing kill comand with –l option.

```
$ kill -l ↵
```

```
1) SIGHUP     2) SIGINT     3) SIGQUIT    4) SIGILL     5) SIGTRAP
6) SIGABRT    7) SIGBUS     8) SIGFPE     9) SIGKILL    10) SIGUSR1
11) SIGSEGV   12) SIGUSR2   13) SIGPIPE   14) SIGALRM   15) SIGTERM
```

You'll find quite a large number of signals.

The most common kill signals are:

Signal Name	Single Value	Effect
SIGHUP	1	Hangup
SIGINT	2	Interrupt from keyboard
SIGKILL	9	Kill signal
SIGTERM	15	Termination signal
SIGSTOP	17, 19, 23	Stop the process

You can use the Signal Value in place of the Signal Name. Therefore, you don't have to memorize all of the names of the various signals.

The syntax of the kill command is:

```
$ kill SIGNAL PID ↵
```

Where SIGNAL is the signal to be sent and PID is the Process ID to be killed.

For e.g., we want to kill the process myprocess which has four running instances with PIDs 3827, 3919, 10764, and 11679 that we got from the *top* command.

So, to send the kill signal, we will issue the commands:

```
$ kill -9 3827 ↵
$ kill -9 3919 ↵
$ kill -9 10764 ↵
$ kill -9 11679 ↵
```

Once we've issued the above commands, all of the myprocess instances will have been successfully killed.

If we already know the process we want to kill is named myprocess, we can make use of the ***killall*** command and send the same signal the process like so:

```
$ killall -9 chrome ↵
```

10.4 BASIC PROCESS ATTRIBUTES AND THEIR ROLE IN ACCESS CONTROL

A process under Linux consists of an address space and a set of data structures in the kernel to keep track of that process. The address space is a section of memory that contains the code to execute as well as the process stack. The kernel must keep track of the following data for each process on the system:

- The address space map
- The current status of the process
- The execution priority of the process
- The resource usage of the process
- The current signal mask
- The owner of the process

10.4.1 Process Attributes

A process has certain attributes that directly affect execution. These include

- **PID** – The PID stands for the Process Identification. This is a unique number that defines the process within the kernel.
- **PPID** – This is the process's Parent PID, the creator of the process.
- **UID** – The User ID number of the user that owns this process.
- **EUID** – The Effective User ID of the process.
- **GID** – The Group ID of the user that owns this process.
- **EGID** – The Effective Group User ID that owns this process.
- **Priority** – The priority that this process runs at.

To view a process you can use the ps command.

```
$ ps -l ↵
```

F S UID PID PPID C PRI NI P SZ:RSS WCHAN TTY TIME COMD
30 S 0 11660 145 1 26 20 * 66:20 88249F10 ttyq 6 0:00 rloginid

- **The F Field:** This is the flag field. It uses hexadecimal values which are added to show the value of the flag bits for the process. For a normal user process this will be 30, meaning it is loaded into memory.
- **The S Field:** The S Field is the state of the process. The two most common values are S (Sleeping) and R (Running). An important value to look for is X, which means the process is waiting for memory to become available.
- **PID Field:** The PID shows the Process ID of each process. This value should be unique. Generally, PID are allocated lowest to highest, but wrap at some point. This value is necessary for you to send signal to a process such as the KILL signal.
- **PRI Field:** This stands for priority field. The process with the lowest value in the PRI field has the highest priority. The lower the value, the higher the priority.
- **P Flag:** This is the processor flag. On the SGI, this refers to the processor the process is running on.

- **SZ Field:** This refers to the SIZE field. This is the total number of pages in the process. Each page is 4096 bytes.
- **TTY Field:** This is the terminal assigned to your process.
- **Time Field:** The cumulative execution time of the process in minutes and seconds.
- **COMD Field:** The command that was executed.

10.4.2 pid and Parentage

Two of the most basic attributes are its process ID, or pid, and the pid of its parent process. A pid is a positive integer that uniquely identifies a running process and is stored in a variable of type pid_t. When a new process is created, the original process is known as the parent of the new process and is notified when the new child process ends.

When a process dies, its exit code is stored until the parent process requests it. The exit status is kept in the kernel's process table, which forces the kernel to keep the process's entry active until it can safely discard its exit status. Processes that have exited and are kept around only to preserve their exit status are known as zombies. Once the zombie's exit status has been collected, the zombie is removed from the system's process table.

If a process's parent exits (making the child process an orphan process), that process becomes a child of the init process. The init process is the first process started when a machine is booted and is assigned a pid of 1. One of the primary jobs of the init process is to collect the exit statuses of processes whose parents have died, allowing the kernel to remove the child processes from the process table.

10.4.3 Credentials

Linux uses the traditional Unix security mechanisms of users and groups. User IDs (uids) and group IDs (gids) are integers that are mapped to symbolic user names and group names through / etc/passwd and /etc/group, respectively. However, the kernel knows

nothing about the names– it is concerned only with the integer representation. The 0uid is reserved for the system administrator, normally called root. All normal security checks are disabled for processes running as root (that is, with a uid of 0), giving the administrator complete control over the system.

In most cases, a process may be considered to have a single uid and gid associated with it. Those are the IDs that are used for most system security purposes (such as assigning ownerships to newly created files).

As Linux developed, restricting processes to a single group turned out to create new difficulties. Users involved in multiple projects had to explicitly switch their gid when they wanted to access files whose access was restricted to members of different groups.

Supplemental groups were introduced in BSD 4.3 to eliminate this problem. Although every process still has its primary gid (which it uses as the gid for newly created files, for example), it also belongs to a set of supplemental groups. Security checks that used to ensure that a process belonged to a particular group (and only that group) now allow access as long as the group is one of the supplemental groups to which the process belongs.

10.5 EXAMINING THE LIST OF RUNNING PROCESSES

Linux provides us with a utility called **ps** for viewing information related with the processes on a system which stands as abbreviation for “Process Status”. ps command is used to list the currently running processes and their PIDs along with some other information depends on different options. It reads the process information from the virtual files in /proc file-system. /proc contains virtual files, this is the reason it’s referred as a virtual file system.

ps provides numerous options for manipulating the output according to our need.

Syntax–

```
ps [options]
```

Options for ps Command:

$ ps	If you run ps command without any arguments, it displays processes for the current shell.
$ ps -A or $ ps -e	Display every active process on a Linux system in generic (Unix/Linux) format.
$ ps au or $ psaxu	Display all processes in BSD format.
$ ps -ef or $ ps -eF	To perform a full-format listing, add the -f or -F flag.
$ ps -x	You can select all processes owned by you.
$ ps -fUtecmint or $ ps -fu 1000	To display a user's processes by real user ID (RUID) or name, use the -U flag.
$ ps -fG apache or $ ps -fG 48	If you want to list all processes owned by a certain group (real group ID (RGID) or name)
$ ps -fp 1178	List Processes by PID

$ ps -t pst/0
$ ps -t pst/1
$ ps -ft tty1
	List Processes by TTY

Process Tree: A process tree shows how processes on the system are linked to each other; processes whose parents have been killed are adopted by the init (or systemd).

```
$ ps -e —forest
```

You can also print a process tree for a given process like this.

```
$ ps -f —forest -C sshd
```

OR

```
$ ps -ef —forest | grep -v grep | grepsshd
```

10.6 BACKGROUND PROCESS

A background process runs without being connected to your keyboard. If the background process requires any keyboard input, it waits.

The advantage of running a process in the background is that you can run other commands; you do not have to wait until it completes to start another.

The simplest way to start a background process is to add an ampersand (&) at the end of the command.

```
$ ls ch*.doc &↵
```

This displays all those files the names of which start with ch and end with .doc –

ch01-1.doc ch010.doc ch02.doc ch03-2.doc
ch04-1.doc ch040.doc ch05.doc ch06-2.doc
ch01-2.doc ch02-1.doc

Here, if the ls command wants any input (which it does not), it goes into a stop state until we move it into the foreground and give it the data from the keyboard.

That first line contains information about the background process - the job number and the process ID. You need to know the job number to manipulate it between the background and the foreground.

Press the Enter key and you will see the following–

[1] + Done ls ch*.doc &
$

The first line tells you that the ls command background process finishes successfully. The second is a prompt for another command.

10.7 JOB CONTROL

In Linux, a job is defined as a task or command that has started running but not yet finished what it is doing. As Linux is a multitasking operating system, it allows you to run multiple commands simultaneously. Each running command is called a job, and is assigned a unique id called the job number.

It is very easy to manage jobs in Linux.

The main commands you use for job control in Linux are as follows.

- **jobs** – List all the jobs that are running or suspended.
- **fg** – Bring the job to the foreground.
- **bg** – Send the job to the background.
- **stop** or **Ctrl + z** – Suspend the job.
- **kill** or **Ctrl + c** – Terminate the job.

10.7.1 Unique Identifiers for a job

You can point out a particular job using special identifiers. They are as follows:

- %n – Job whose job number / job ID is the number n
- %word – Job whose command line starts with string word
- %?word – Job whose command line contains the string word
- %+ – Current job. You can also use %%
- %– – Previous job

10.7.2 Managing jobs in Linux

Managing jobs in Linux can be easily understood via an example. Let us simultaneously run a few commands that takes some time to complete.

```
$ sleep 150 &
$ sleep 250 &
$ sleep 300 &
$ sleep 200 &
```

Here we have executed 4 sleep commands and all are started in the background as denoted by &.

List the running jobs

```
$ jobs
```

And the output is ...

```
[1]    running    sleep 150
[2]    running    sleep 250
[3]  - running    sleep 300
[4]  + running    sleep 200
```

In the output above, the number within [and] is the job number (job ID). The job number is unique for each job.

Bring a job to the foreground

```
$ fg %job-number
```

To bring job number 2 to the foreground, you can run the following command.

```
$ fg %2
```

Suspend the job: To suspend a job, you first bring the job to the foreground and then press the keys Ctrl + z.

Alternately, you can also suspend a job running in the background as follows.

```
$ stop %job-number
```

Send a job to the background

```
$ bg %job-number
```

To resume the suspended job 2 in the background, you can run the following command.

```
$ bg %2
```

List only running jobs

```
$ jobs -r
```

List only suspended jobs

```
$ jobs -s
```

List process IDs of the jobs: This command will list process IDs of the jobs in addition to the normal information.

```
$ jobs -l
```

Terminate a job: To terminate a job, you first bring the running job to the foreground, and then press the keys Ctrl + c. You can also terminate a job without bringing it in the foreground just by passing the job number to kill.

```
$ kill %job-number
```

For example, to kill the 3rd job, you can do as follows.

```
$ kill %3
```



CHECK YOUR PROGRESS

Q.1: PID stands for .....

.....

Q.2: Which command is used to list all the jobs that are currently running or suspended?

.....

Q.3: How will you run a process in the background?

.....

10.8 CRONTAB FILE FORMAT

The crontab is a list of commands that you want to run on a regular schedule, and also the name of the command used to manage that list. Crontab stands for “cron table’’, because it uses the job scheduler cron to execute tasks; cron itself is named after “chronos’’, the Greek word for time. Cron is the system process which will automatically perform tasks for you according to a set schedule. The schedule is called the crontab, which is also the name of the program used to edit that schedule.

10.8.1 crontab File Fields

The Linux Crontab Format is:

```
MIN HOUR DOM MON DOW CMD
```

The Crontab Fields and Allowed Ranges (Linux Crontab Syntax)

Field	Description	Allowed Value
MIN	Minute field	0 to 59
HOUR	Hour field	0 to 23
DOM	Day of Month	1 to 31
MON	Month field	1 to 12
DOW	Day Of Week	0-6
CMD	Command	Any command to be executed.

Example 10.1: Scheduling a Job For a Specific Time

The basic usage of cron is to execute a job in a specific time as shown below. This will execute the Full backup shell script (full-backup) on 10th June 08:30 AM.

The time field uses 24 hours format. So, for 8 AM use 8, and for 8 PM use 20.

30 08 10 06 * /home/maverick/full-backup

30 – 30th Minute

08 – 08 AM

10 – 10th Day

06 – 6th Month (June)

* – Every day of the week

Example 10.2: To view the Crontab entries

a) View Current Logged-In User's Crontabentries: To view your crontab entries type crontab -l from your unix account.
b) View Root Crontab entries: Login as root user (su – root) and do crontab -l.
c) To view crontab entries of other Linux users: Login to root and use -u {username} -l.

Example 10.3: To edit Crontab Entries

Edit Current Logged-In User's Crontab entries.

To edit a crontab entries, use crontab -e. By default this will edit the current logged-in users crontab.

Example 10.4: To schedule a job for every minute using Cron

Ideally you may not have a requirement to schedule a job every minute. But understanding this example will will help you understand the other examples.

* * * * * CMD

The * means all the possible unit– i.e., every minute of every hour through out the year. More than using this * directly, you will find it very useful in the following cases.

When you specify */5 in minute field means every 5 minutes.

When you specify 0-10/2 in minute field mean every 2 minutes in the first 10 minute.

Thus the above convention can be used for all the other 4 fields.

Example 10.5: To schedule a job for more than one time (e.g., Twice a Day)

The following script take a incremental backup twice a day every day.

This example executes the specified incremental backup shell script (incremental-backup) at 11:00 and 16:00 on every day. The comma separated value in a field specifies that the command needs to be executed in all the mentioned time.

00 11, 16 * * * /home/maverick/bin/incremental-backup

00 – 0th Minute (Top of the hour)

11, 16 – 11 AM and 4 PM

* – Every day

* – Every month

* – Every day of the week

Example 10.6: To schedule a job for certain range of time (e.g., Only on Weekdays)

If you wanted a job to be scheduled for every hour with in a specific range of time then use the following.

Cron Job everyday during working hours: This example checks the status of the database everyday (including weekends) during the working hours 9 a.m – 6 p.m

00 09-18 * * * /home/maverick/bin/check-db-status

00 – 0th Minute (Top of the hour)

09-18 – 9 am, 10 am, 11 am, 12 am, 1 pm, 2 pm, 3 pm, 4 pm, 5 pm, 6 pm

* – Every day

* – Every month

* – Every day of the week

Cron Job every weekday during working hours: This example checks the status of the database every weekday (i.e excluding Sat and Sun) during the working hours 9 a.m – 6 p.m.

00 09-18 * * 1-5 /home/maverick/bin/check-db-status

00 – 0th Minute (Top of the hour)

09-18 – 9 am, 10 am, 11 am, 12 am, 1 pm, 2 pm, 3 pm, 4 pm, 5 pm, 6 pm

* – Every day

* – Every month

1-5 – Mon, Tue, Wed, Thu and Fri (Every Weekday)

Example 10.7: To schedule a background Cron job for every 10 minutes.

Use the following, if you want to check the disk space every 10 minutes.

*/10 * * * * /home/maverick/check-disk-space

It executes the specified command check-disk-space every 10 minutes through out the year.

10.9 BACKUP AND RESTORE PROCEDURE

UNIX and LINUX backup and restore can be done using backup commands tar, cpio, dump and restore. Any backup solution using these commands depends on the type of backup you are taking as these backup commands offers different capabilities.

10.9.1 Using tar Command

The features of the tar command are:

- tar is used for single or multiple files backup and restore on/from a tape or file.
- Tar can not backup special character & block device files, shows as 0 byte files with first letter of permissions as b or c for block or character.
- tar Works only on mounted file system, it can not access the files on unmounted file system.

Following is a list of common command line switches and options used with tar:

Switch	Action
-c	Creates a new .tar archive
-C	Extracts to a different directory
-j	Uses bzip2 compression
-z	Uses gzip compression
-v	Verbose show archiving progress
-t	Lists archive contents
-f	File name of the archive
-x	Extracts tar archive

Backing up all files in a directory including subdirectories to a tape device (/dev/rmt/0) or a file.

Example 10.8:

```
$tar –cvf /dev/rmt/0 *
```

In the command above Options are c –> create ; v –> Verbose ; f–>file or archive device ; * –> all files and directories . Together the commands means create a tar file on /dev/rmt/0 from all file and directories s in the current directory.

Example 10.9:

```
$tar - cvf /home/backup *
```

Create a tar called backup in home directory , from all file and directories s in the current directory.

Viewing a tar backup on a tape or file: t option is used to see the table of content in a tar file.

```
$tar -tvf    /dev/rmt/0  ## view files  backed up
on a tape device.
$tar tvf  /home/backup  ## view files  backed up
inside the  backup
```

In the command above options are c –> create ; v –> Verbose ; f–>file or archive device ; * –> all files and directories. Together the commands means create a tar file on /dev/rmt/0 from all file and directories s in the current directory.

Extracting tar backup from the tape: x option is used to extract the files from tar file.

```
$tar -xvf /dev/rmt/0        ##extract /  restore
files in to current directory.
$tar xvf /home/backup ## extract / restore files
in to current directory.
```

Note: Restoration will go to present directory or original backup path depending on relative or absolute path names used for backup.

10.9.2 Using cpio Command

The features of cpio command are:

- Used for single or multiple files backup.
- Can backup special character & block device files.
- Works only on mounted file system.
- Need a list of files to be backed up.
- Preserve hard links and time stamps of the files.

Using cpio command to backup all the files in current directory to tape.

```
find . -depth -print | cpio -ovcB> /dev/rmt/0
```

cpio expects a list of files and find command provides the list, cpio has to put these file on some destination and a > sign redirect these files to tape. This can be a file as well .

Viewing cpio files on a tape:

```
cpio -ivtB< /dev/rmt/0    ## Options i -> input ;
v->verbose; t-table of content; B-> set I/O block
size to 5120 bytes
```

Restoring a cpio backup:

```
cpio -ivcB< /dev/rmt/0  ## Options i -> input ; v-
>verbose; t-table of content; B-> set I/O block
size to 5120 bytes
```



CHECK YOUR PROGRESS

Q.4: How will you kill a process?

.....

Q.5: cpio command works only on mounted file systems. (TRUE or FALSE)

Q.6: Which command line switch is used with tar command to extract a tar archive?

.....



10.10 LET US SUM UP

- A process refers to a program in execution. Whenever a command is issued in Linux, it creates/starts a new process.
- Linux keeps account of the processes through a 5 digit ID number known as Process ID or PID.
- Processes can be killed using their PID or name.
- A process under Linux consists of an address space and a set of data structures in the kernel to keep track of that process.
- A process has certain attributes that directly affect execution. They are– PID (Process Identification), PPID (Parent PID), UID (User ID), EUID (Effective User ID), GID (Group ID), EGID (Effective Group User ID) and Priority of the process.
- To view a process you can use the ps command.
- A process tree shows how processes on the system are linked to each other.
- The simplest way to start a background process is to add an ampersand (&) at the end of the command.
- In Linux a job is defined as a task or command that has started running but not yet finished what it is doing.
- The crontab is a list of commands that you want to run on a regular schedule, and also the name of the command used to manage that list.

- Unix and Linux backup and restore can be done using backup commands tar, cpio, dump and restore. Any backup solution using these commands depends on the type of backup you are taking as these backup commands offers different capabilities.



10.11 FURTHER READING

1) Das, S. (2012). *Your UNIX/*Linux: *The Ultimate Guide*. (Third Edition). (Science/Engineering/Math). McGraw-Hill.
2) Negus, C. (2012). *Linux Bible*. (Vol. 772). John Wiley & Sons.



10.12 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: Process Identification.

Ans. to Q. No. 2: jobs

Ans. to Q. No. 3: For running a process in background, use "&"

Ans. to Q. No. 4: Using the kill command.

Ans. to Q. No. 5: TRUE

Ans. to Q. No. 6: The –x option.



10.13 MODEL QUESTIONS

Q.1: What is a process? What are the types of process in Linux?

Q.2: Explain process attributes. Which command is used to view process attributes?

Q.3: Explain ps command with various options.

Q.4: Describe background process in Linux.

Q.5: What are the commands used for job control. Give examples.

Q.6: Explain crontab command. How can you schedule a job for every minute using crontab?

Q.7: Describe the features of tar command with options.

*** ***** ***

UNIT 11: PRINTING JOBS

UNIT STRUCTURE

11.1 Learning Objectives

11.2 Introduction

11.3 Submitting a Print Job

11.4 Managing a Print Job

- 11.4.1 Listing Jobs on the Print Queue
- 11.4.2 Controlling the Printers
- 11.4.3 Cancelling or Removing Print Job

11.5 Checking Status of a Print Job

11.6 Configuring the Print Queue

11.7 Selecting and Configuring Print Driver

11.8 Editing the Printer Configuration

11.9 Let Us Sum Up

11.10 Further Reading

11.11 Answers To Check Your Progress

11.12 Model Questions

11.1 LEARNING OBJECTIVES

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

- print documents in Linux
- check the status of a printing job
- to cancel or remove the printing of a document
- see the list of the print jobs on the queue
- to select a print driver for printing documents
- to set the printer configuration.

11.2 INTRODUCTION

In the previous unit, you have learnt about the processes, different phases of a process in Linux system, how to create processes and kill processes (i.e. stopping a process) by using different commands. You have

also learnt about the background processes and how to examine running processes and the status of the processes. In addition to these you have been acquainted with topics such as the controlling of jobs in Linux, how to back up data and files and how data and files can be restored in Linux.

Linux has different methods that can be used to perform the printing jobs. In this unit, you will learn how to submit a document for printing, how to print documents, examine the print queue, remove jobs from the print queue and configure your printing environment, check the status of printing job, see the list of the print jobs on the queue, select a print driver etc. In the next unit, we will learn about Linux user, creating and managing accounts etc.

11.3 SUBMITTING A PRINT JOB

The **lpr** command is used for submitting a print job to the printer. The syntax for writing the lpr command is as follows-

```
$ lpr   filename
```

where *filename* is the name of the file to be printed.

$ lpr * means that not more than 30 files are specified. If more than 30 files are specified, then nothing will be printed and no notification of the failure is displayed.

The lpr command communicates with **lpd**, which is called the spooling daemon. Whenever a request comes in (e.g. from the lpr command), the printer daemon accepts the print data and options. When lpd identifies an available printer for a queue with a pending job it spawns a child lpd process to handle the printing. The child lpd daemon applies any appropriate filters as defined in /etc/printcap before sending the data to the printer.

The *lpr* command has several command-line arguments.The most widely used arguments are: -**P** printer specifies the printer to use, -**h** suppresses printing of the burst page, -**s** creates a symbolic link instead of copying the file to the spool directory (useful for large files), and -**#**num specifies the number of copies to print.

For example,

```
$ lpr -Pxrx_233 abc.txt
```

Here, *abc.txt* is the file to be printed and **xrx_233** is the name of the printer.

```
$ lpr -#2 -sPdj abc.txt
```

will create a symbolic link to the file abc.txt in the spool directory for the printer named *dj*, where it would be processed by *lpd*. It would then print a second copy of abc.txt.

11.4 MANAGING A PRINT JOB

The commands **lpq**, **lpc** and **lprm** are used to manage the print jobs.

11.4.1 Listing Jobs on the Print Queue

The **lpq** command lists the jobs in the print queue. If run without any parameters it will display jobs on the default queue. The most common options are:

`lpq -a` : It lists jobs on all printers

`lpq -l` : It provides more information about the jobs

11.4.2 Controlling the Printers

The **lpc command** is used to manage the print queue including (stopping the printer, starting the printer and reordering the jobs on the print queue).

For example,

lpc stop *printername*

Stops the spooling daemon after the current job has completed. It then disables printing to that printer. The printername can be replaced with the keyword all to view the status of all local printers.

lpc start *printername*

Enables printing starting a spooling daemon for the printer. The printername can be replaced with the keyword all to view the status of all local printers.

11.4.3 Cancelling or Removing Print Jobs

To remove a print job the **lprm** command is used. It can remove one or more jobs from the print queue. The command can be run without any arguments which will delete the current print request. Some examples of the lprm command are given below.

```
lprm -P printqueue jobid
```

Remove requests from the specified queue.

```
lprm -
```

If this command is requested by a user then it will remove all jobs from that user and if it is specified by a superuser then it will remove all requests on that print queue.

```
lprm -a
```

Removes all print jobs on all printers

```
lprm -a all
```

Removes all jobs on all printers from all users. This can be performed by only superuser.

11.5 CHECKING STATUS OF A PRINT JOB

The command **lpstat** displays status information about the current classes, jobs, and printers. When this command is run without any argument, then it lists jobs queued by the current user. *lpstat* has a number of options that provides the status information of printers and print jobs.

Some commonly used options are:

-a [*printer(s)*] : Shows the accepting state of printer queues. If no printers are specified then all printers are listed.

-c [*class(es)*] : Shows the printer classes and the printers that belong to them. If no classes are specified then all classes are listed.

-d : Shows the current default destination.

-l : Shows a long listing of printers, classes, or jobs.

-o [*destination(s)*] : Shows the jobs queue on the specified destinations. If no destinations are specified all jobs are shown.

-p [*printer(s)*] : Shows the printers and whether or not they are enabled for printing. If no printers are specified then all printers are listed.

-r : Shows whether the CUPS server is running.

-s : Shows a status summary, including the default destination, a list of classes and their member printers, and a list of printers and their associated devices. This option is equivalent to using the "**-d**", "**-c**", and "**-v**" options.

-t : Shows all status information. This option is equivalent to using the "**-r**", "**-d**", "**-c**", "**-v**", "**-a**", "**-p**", and "**-o**" options.

-u [*user(s)*] : Shows a list of print jobs queued by the specified users. If no users are specified, lists the jobs queued by the current user.

-v [*printer(s)*] : Shows the printers and the attached devices. If no printers are specified then all printers are listed.



CHECK YOUR PROGRESS

Q.1: Which command is used for submitting and printing a file in Linux?

.....

Q.2: State the functions of lpc and lpq commands.

.....

.....

Q.3: Which function is used for cancelling a print job?

.....

.....

Q.4: Without any argument ..... command lists the jobs in the default printers queue.

11.6 CONFIGURING THE PRINT QUEUE

Setting up a printer in Linux involves setting up a queue for the print spooler or scheduler-the process that accepts and sends files for printing and does whatever is needed to get them printed. The queue is essentially a holding place for files that are to be sent to a printer. Before being sent to the printer, however, the files are usually processed to convert them to a form that the printer can understand.

The printer daemon sends the file to be printed to the standard input of the filter, and the filter modifies the file as needed. The filter can then write the modified file to standard output, which the print system sends on to the printer.

The capabilities of a particular printer are defined in the **/etc/printcap** file, which is normally created when Linux is installed. Each printer or printer configuration that you use must be defined in this file. You can set up multiple configurations for the same printer to handle different file types, and the print system can send output to other devices or save it to a file.

Each printer configuration in *printcap* file defines a print queue. Linux applications use the **lpr** command to make the actual print request by using the line-printer daemon, **lpd**, which spawns a copy of itself to handle each print request. **lpd** is the main program and runs continuously in the background to handle all requests from lpr, **lpq** (for querying print job and printer status)

Common entries in the printcap file are:

- **lp** – Local printer port (used for printing by local printer)
- **sd** – Spool directory
- **mx** – Maximum buffer size (0 means unlimited)
- **sh** – suppress printing of burst page header
- **rm** – remote server host name (only for remote printing)
- **rp** – remote printer queue name (only for remote printing)
- **if** – input filter, used to process the input with a program to change the formatting.
- **af** – accounting file
- **lf** – error log file

pl – page length (lines)

pw – page width (characters)

Each entry defines a printer queue and consists of a series of fields delimited with the colon (:). The first field of the entry supplies one or more names for the print queue, separated by the pipe (|) character. The entry "lp" should be used as the name of one of the print queues, since it will be the default queue when print without specifying a queue. The last sub-field of this first field should contain a description of this print queue and may be displayed by some print management software.

The following two examples show the format for the printcap file.

For local printer say **printer-_1**

```
printer_1|localprinter:\
        :lp=/dev/lp0:\
        :sd=/var/spool/lpd/printer_1:\
        :mx#0:\
        :sh:
```

For remote printer say **printer_2**

```
printer_2|remoteprinter:\
        :rm=remoteserver:\
        :rp=remoteprinter:\
        :sd=/var/spool/lpd/printer_2:\
        :mx#0:\
        :sh:
```

11.7 SELECTING AND CONFIGURING PRINT DRIVER

Linux includes several configuration tools for print driver that can be used in a graphical or command line environment. Additionally, the configuration tools included can be administered remotely for added convenience if issues arise while you are away.

The **Printer Configuration Tool** is a utility for Linux that allows administrators to add, edit, and configure print driver to work in a local or networked environment. Administrators can use the **Printer Configuration Tool** on local printers attached to workstations, and even share such printers

to other users on the network. The **Figure 11.1** shows the **Printer Configuration Tool** in action.

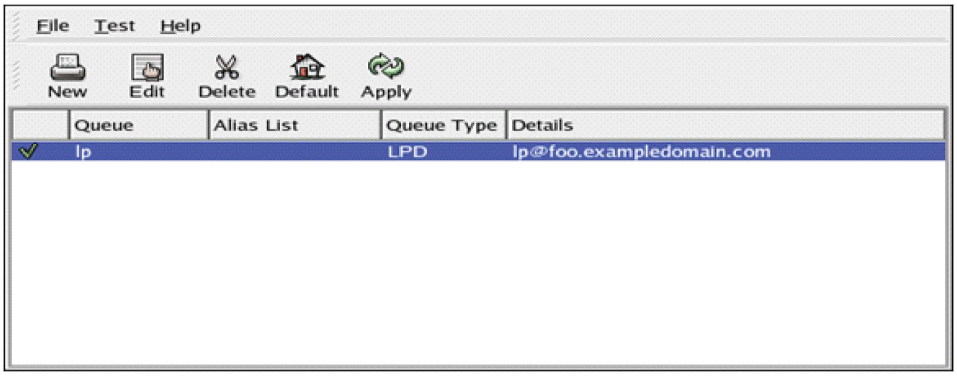


Figure 11.1: The Printer Configuration Tool

Adding a New Printer: To add a new printer the following steps are performed.

1) Click the **New** button and type a queue name.
2) Choose a printer **Queue Type** from the list.
3) You have the option to select the **Printer Device** shown, rescan your devices for the correct device, or create your own custom device. This will be the **/dev** entry that represents the interface between the device driver and the printer itself. In most instances, the device will be named **/dev/lp0**.
4) Now you can choose a **Printer Driver** by selecting one from the extensive list. Choose the printer make and model, and select the driver. Drivers marked with an asterisk (*) are recommended drivers. If you are configuring a remote printer or a printer that does not have a corresponding driver for its make and model, the safest choice would be **Postscript Printer**.

Once the printer and print server is installed and configured properly, you should configure client computers to access the print server.

Any post-install configuration on printers or the print server can be done via the Printer Configuration Tool by clicking **Edit** and choosing the **Driver Options**.

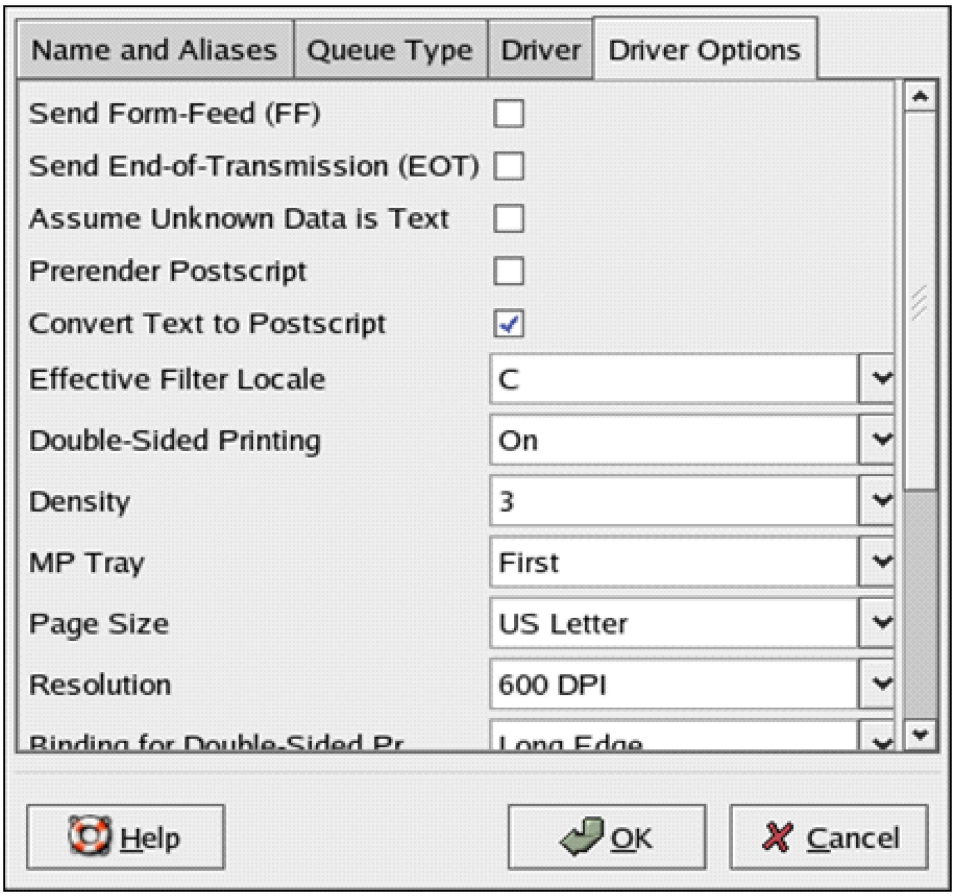


Figure11.2: Driver Options



CHECK YOUR PROGRESS

Q.5: Which file is related to printer configuration and defining the print queue?

.....

.....

Q.6: Which is the most preferred and safest printer driver?

.....

11.8 EDITING THE PRINTER CONFIGURATION

The printer settings can be changed after adding a printer. To edit the settings you need to select the printer list of the Printer Configuration tool and then clicking the Edit button. The tabbed window called “Edit a print queue” appears. The window contains the current values for the printer that

has been selected to edit. Make any changes you need then click OK. Then click Apply in the main Printer Configuration tool window to save the changes and restart the printer daemon.

The tabs in the tabbed window are:

- **Queue Name:** To rename a printer, change the value of Name on the Queue Name tab. Then click OK to return to the main window. The name of the printer changes in the printer list. Click Apply to save the change and restart the printer daemon.
- **Queue Type:** The Queue Type tab shows the queue type that you selected when adding the printer and its settings. You can change the queue type of the printer or just change the settings. After making modifications, click OK to return to the main window. Click Apply to save the changes and restart the printer daemon. Depending on which queue type you choose, you will see different options.
- **Queue Options:** From the Queue Options tab, you can select banner pages before and after your print job. You can also set the printable area of the page. To modify filter options, highlight the option and click Edit to modify or click Delete to remove it. Click OK to accept your changes and return to the main window. Click Apply to save the change and restart the printer daemon.
- **Printer Driver:** The Printer Driver tab shows which print driver is currently being used. This is the same list that you use when you add the printer. If you change the print driver, click OK to accept your changes and return to the main window. Then click Apply to restart the printer daemon.
- **Driver Options:** The Driver Options tab displays the advanced printer options that are different for different printers. Commonly found options are:
  - *Send Form-Feed (FF):* Selecting this options forces the last page out of the printer. If this option does not force the last page out of the printer, try selecting Send End-of-Transmission (EOT) instead. Some printers require both Send Form-Feed (FF) and Send End-of-Transmission (EOT) to eject the last page.

- *Send End-of-Transmission (EOT):* This option is selected if sending a form feed does not work. Refer to the preceding bullet on the Send Form-Feed (FF) option.
- *Assume Unknown Data Is Text:* If this option is selected, the print driver assumes that any data it cannot recognize is text and tries to print it as text. You can select this option if you are having problems in printing.
- *Prerender PostScript:* You can select this option if you are trying to print characters outside of the basic ASCII character set that won't print correctly.
- *Convert Text to PostScript*: This option is selected by default. If your printer can print plain text, try deselecting this when printing plain text documents to decrease the time it takes to print.
- *Page Size:* It allows to select the paper size for the printer, such as US Letter, US Legal, A3, and A4.
- *Effective Filter Locale:* Sets the defaults to C. If you are printing Japanese characters, select ja_JP. Otherwise, accept the default of C.
- *Media Source:* Select this option to use paper from a different tray.

After modifying the driver options, click OK to return to the main window. Then click Apply to save the changes and restart the printer daemon.

To remove an existing printer from the printer list, select the printer and click the Delete button on the toolbar. Then the printer will be removed from the printer list. Click Apply to save the changes and restart the printer daemon.

To set the default printer, select the printer from the printer list and click the Default button on the toolbar. The default printer icon appears in the first column of the printer list beside the default printer.



11.9 LET US SUM UP

- The submission and printing job is done in association with *lpr* command.
- The *lpq* command works with different options to lists the jobs in the print queue.

- The *lpc* command is used to stop a printer, start a printer and reordering the jobs on the print queue.
- The *lprm* command performs the cancelling of the print job from the print queue with different options.
- *lpstat* command is used to display the status of a printer.
- The printer configuration is done from the *printcap* file. Several entries are found under the *printcap* file.
- *Printer Configuration Tool* is a utility for selecting and configuring a printer in Linux system. This tool also used for editing a printer configuration.



11.10 FURTHER READING

1) Das, S. (2012). *Your UNIX/*Linux: *The Ultimate Guide*. (Third Edition). (Science/Engineering/Math). McGraw-Hill.
2) Negus, C. (2012). *Linux Bible*. (Vol. 772). John Wiley & Sons.



11.11 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: The lpr command is used for submitting and printing a file.

Ans. to Q. No. 2: The **lpc** command stops a printer, starts a printer and reorders the print jobs on the print queue. The **lpq** command lists the print jobs in the print queue.

Ans. to Q. No. 3: The **lprm** command is used for cancelling a print job from the print queue.

Ans. to Q. No. 4: Without any argument **lpq** command lists the jobs in the default printer's queue

Ans. to Q. No. 5: The file related to printer configuration and defining the print queue is printcap.

Ans. to Q. No. 6: Postscript is the preferred and safest printer driver.



11.12 MODEL QUESTIONS

Q.1: Briefly describe the process of submitting a print job.

Q.2: Describe the options used with the **lpstat** command.

Q.3: How is the configuration of a printer and print queue is done?

Q.4: Name the commonly found tabs in the Edit a Print Queue window.

Q.5: Give the steps for adding a new printer.

*** ***** ***

UNIT 12: LINUX USERS

UNIT STRUCTURE

12.1 Learning Objectives

12.2 Introduction

12.3 Root Account

12.4 Becoming Superuser

12.5 Managing User Accounts

- 12.5.1 Create New User Accounts
- 12.5.2 Modify User Accounts
- 12.5.3 Delete User Accounts

12.6 Changing Password

12.7 System Monitoring and Logging

12.8 Monitoring Memory Usage and Disk Space Usage

12.9 Let Us Sum Up

12.10 Further Reading

12.11 Answers to Check Your Progress

12.12 Model Questions

12.1 LEARNING OBJECTIVES

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

- describe about the root account of Linux system
- explain how to become superuser
- explain how to manage user accounts
- explain how to change user account password
- describe different system monitoring and logging commands
- explain how to monitor memory usage and disk space usage.

12.2 INTRODUCTION

In the previous unit we have learned about printing and managing jobs. We have also learned about selecting and configuring the print driver. In this unit we will learn about ‘root’ account of Linux system and the role of

root account. We will also learn about how to become **superuser**. We will discuss how to manage user accounts by creating new user accounts, by modifying user accounts or by removing an existing account in this unit. In this unit we will also discuss about how to change the account password and different commands for system monitoring and logging are also explained along with monitoring memory usage and disk space usage. In the next unit, we will explore the concept of networking in Linux in detail.

12.3 ROOT ACCOUNT

Root account is also known as superuser. Root account has complete and full control of the system. A superuser is the most privileged user on the Linux system that able to run any commands without any restriction. This user should be assumed as a system administrator. The root user can do many things which an ordinary user cannot, such as installing new software, changing the ownership of files, and managing other user accounts.

Root account should not be used to perform ordinary tasks, such as web browsing, document writing etc. A simple mistake can cause a big problem to the entire system. Root has the ability to modify the system in any way desired and to grant and revoke the access permissions for the other users.

A critical means for preventing users from directly damaging Unix-like systems or increasing the vulnerability of such systems to damage by others is the avoidance of using the root account except when it is absolutely necessary, even by knowledgeable and experienced system administrators. That is, rather than routinely logging into the system as root, administrators should log in with their admin account and then use the ***su*** command. This will provide us with root privileges only as needed and without requiring a new login.

12.4 BECOMING SUPERUSER

Superuser refers to root user who have the privilege of unrestricted access of all files and commands. In Linux, there are two ways to become a superuser. The first way is to directly log in as root and become superuser

and the second way is to execute the ***su*** command from other user account. The ***su*** command may be used to change one's current account to that of a different user after entering the proper password. It takes the username corresponding to the desired account as its argument; *root* is the default when no argument is provided.

Syntax for ***su*** command to change the current account is

```
$ su username
```

After executing this command the system prompts for the account password.

If we execute the ***su*** command without the argument then system prompted for the **root** password. If we type the password correctly, we will get the normal root account prompt (by default, a hash sign: #), indicating that we have successfully become superuser and that the rules normally restricting file access and command execution do not apply. If we type wrong password then it will display an error message and return to the normal command prompt.

With the help of **exit** command or **Ctrl+D** we may exit from the superuser account. We may suspend the shell and place it in the background with the **suspend** command.

12.5 MANAGING USER ACCOUNT

With the help of **su** we can manage user accounts by creating new user, delete existing user or modify the user.

12.5.1 Create New User Accounts

We can create new user by using the ***useradd*** command.

```
$ useradd username
```

The above command create a new user account named ***username*** and adds an entry to the **/etc/passwd, /etc/shadow, /etc/group** and **/etc/gshadow** files.

To be able to log in as the newly created user, we need to set the user password. We can do that by using the command ***passwd*** followed by the username.

```
$ passwd username
```

It will prompt to enter and confirm the password:

Output:

Changing password for user username:

New password:

Retype new password:

passwd: all authentication tokens updated successfully

Add a new user and create home directory: While creating a new user account using the ***useradd*** command the user home directory is not created.

Use the -m option to create the user home directory as

```
$ useradd -m username
```

The command above creates the new user's home directory, and copies files from /etc/skel directory to the user's home directory.

Creating a user with specific home directory: To create a user's home directory other than default /home directory use -d option.

Suppose we want to create a user ***uername*** in the home directory ***myhome*** then execute the following command.

```
$ useradd -m -d /opt/myhome username
```

Creating a user with specific user ID: In LINUX users are identified by username and unique identifier (UID).

User identifier (UID) is a unique positive integer assigned by the Linux system to each user. The UID along with other access control policies is used to determine the types of actions a user can perform on system resources.

By default when a new user is created the system assigns the next available UID from the range of user IDs specified in the **login.defs** file.

Use -**u** option to create a user with specific UID. For example if we want to create a user ***username*** with the UID 1000 then run the following command.

```
$ useradd –u 1000 username
```

We can verify the UID by using following **id** command.

```
$ id –u username
```

Output:

1000

12.5.2 Modify User Account

Superuser can modify other user accounts with **usermod** command. The **usermod** command is simple to use with lots of options to make changes to an existing user.

Syntax of **usermod** is as follows:

```
$ usermod [option] username
```

The following options can be used with **usermod** command.

-a, **–append:** Add the user to the supplementary group(s). Use only with the **-G** option.

-c, **–comment:** The new value of the user's password file comment field. It is normally modified using the **chfn(1)** utility.

-d, **–home** *HOME_DIR:* The user's new login directory.

-e, **–expiredate** *EXPIRE_DATE:* The date on which the user account will be disabled. The date is specified in the format *YYYY-MM-DD*. An empty *EXPIRE_DATE* argument will disable the expiration of the account. This option requires a /etc/shadow file. A /etc/shadow entry will be created if there were none.

-f, **–inactive** *INACTIVE:* The number of days after a password expires until the account is permanently disabled. A value of 0 disables the account as soon as the password has expired, and a value of –1 disables the feature. This option requires a /etc/shadow file. A /etc/shadow entry will be created if there were none.

-g, **–gid** *GROUP:* The group name or number of the user's new initial login group. The group must exist. Any file from the user's home directory owned by the previous primary group of the user will be owned by this new group. The group ownership of files outside of the user's home directory must be fixed manually.

-G, **–groups** *GROUP1*[,*GROUP2,...*[,*GROUPN*]]]: A list of supplementary groups of which the user is also a member. Each group is separated from the next by a comma, with no intervening whitespace. The groups are subject to the same restrictions as the group given with the **-g** option. If the user is currently a member of a group which is not listed, the user will be removed from the group. This behavior can be changed via the **-a** option, which appends the user to the current supplementary group list.

-l, **–username** *NEW_username:* The name of the user will be changed from ***username*** to *NEW_username*. Nothing else is changed. In particular, the user's home directory or mail spool should probably be renamed manually to reflect the new login name.

-L, **–lock:** Lock a user's password. This puts a '!' in front of the encrypted password, effectively disabling the password. You can't use this option with **-p** or **-U**.

-m, **–move-home:** Move the content of the user's home directory to the new location. This option is only valid in combination with the **-d** (or **–home**) option.



CHECK YOUR PROGRESS

Q.1: Fill in the blanks:

i) Root account is also known as .....

ii) ..... command is used to exit from superuser.

iii) ..... command is used to create a new user account.

iv) Superuser can modify other user accounts with ..... command.

12.5.3 Deleting User Account

Superuser can delete an existing user account with **userdel** command.

Syntax: `$ userdel [option] username`

The following options can be used with **userdel** command.

-f, –force : This option will delete the account forcefully, even if user is still logged in.

-r, –remove: This option will delete the user account along with the home directory.

-R, –root *CHROOT_DIR*: Apply changes in the *CHROOT_DIR* directory and use the configuration files from the *CHROOT_DIR* directory.

12.6 CHANGING PASSWORD

A user can change password of own account. But superuser can change the password of any user accounts. p**asswd** command is used to change the account password.

Syntax: `$ passwd [option] username`

The user is first prompted for his/her old password, if one is present. This password is then encrypted and compared against the stored password. The user has only one chance to enter the correct password. The superuser is permitted to bypass this step so that forgotten passwords may be changed.

After the password has been entered, password aging information is checked to see if the user is permitted to change the password at this time. If not, **passwd** refuses to change the password and exits.

The following options can be applied to **passwd** command.

-a, –all : This option can be used only with **-S** and causes show status for all users.

-d, –delete : This option will delete an existing password of an account.

-e, –expire : It will immediately expire an account password.

-i, –inactive : This will inactive an account after the expiry of password.

12.7 SYSTEM MONITORING AND LOGGING

In LINUX, there are several commands to monitor the system's performance.

Following are some **system monitoring** commands:

- ***top*: *top*** command is a monitoring command which is used to monitor the system performance. It displays CPU usage, Memory usage, Swap Memory, Cache Size, Buffer Size, Process PID, User, Commands and much more.
- ***vmstat- Virtual Memory Statistics*: *vmstat*** command is used to display statistics of virtual memory, kernel threads, disks, system processes, I/O blocks, interrupts, CPU activity and much more.
- ***lsof (List open files): lsof*** command is used to display the list of open files and processes.
- ***tcpdump: tcpdump*** is a network packet analyzer. This command is used to capture or filter TCP/IP packets that received or transferred on a specific interface over a network.
- ***netstat: netstat*** is a command line tools used for monitoring the incoming and outgoing network packets statistics and interface statistics.

System Logging: The powerful logging system of LINUX enables us to record almost anything we imagine and then manipulate the logs to retrieve the information we require.

Several versions of UNIX provide a general-purpose logging facility called **syslog.**

UNIX *syslog* is a host-configurable, uniform system logging facility *which is also used in LINUX*. The system uses a centralized system logging process that runs the program **/etc/syslogd** or **/etc/syslog**.

The operation of the system logger is quite straightforward. Programs send their log entries to *syslogd*, which consults the configuration file **/etc/syslogd.conf** or **/etc/syslog** and, when a match is found, writes the log message to the desired log file.

We should understand the four basic ***syslog*** terms:

- **Facility:** The identifier used to describe the application or process that submitted the log message. For example, mail, kernel, and ftp.

- **Priority:** An indicator of the importance of the message. Levels are defined within ***syslog*** as guidelines, from debugging information to critical events.
- **Selector:** A combination of one or more facilities and levels. When an incoming event matches a selector, an action is performed.
- **Action:** What happens to an incoming message that matches a selector– Actions can write the message to a log file, echo the message to a console or other device, write the message to a logged in user, or send the message along to another **syslog** serve.

12.8 MONITORING MEMORY USAGE, DISK SPACE USAGE

In Linux there are commands for monitoring the memory usages and disk space usages. Following are some commands with the help of which we can monitor the memory usage.

1) ***free*** command: **free** command is a command which displays the total amount of free space available along with the amount of memory used and swap memory in the system, and also the buffers used by the kernel.

   Syntax: ***$ free [option]***

   Following options can be used with free command.

   - **-b, – -bytes:** It displays the memory in bytes.
   - **-k, – -kilo:** It displays the amount of memory in kilobytes (default).
   - **-m, – -mega:** It displays the amount of memory in megabytes.
   - **-g, – -giga:** It displays the amount of memory in gigabytes.
   - **-h, – -human:** It shows all output columns automatically scaled to shortest three digit unit and display the units also of print out. The units used are **B**(bytes), **K**(kilos), **M**(megas), **G**(gigas), and **T**(teras).
   - **-c, – -count:** It displays the output **c** number of times and this option actually works with **-s** option.
   - **-l, – -lohi:** It shows the detailed low and high memory statistics.

- **-o, – -old:** This option disables the display of the buffer adjusted line.
- **-s, – -seconds:** This option allows you to display the output continuously after **s** seconds delay. In actual, the **usleep** system call is used for microsecond resolution delay times.
- **-t, – -total:** It adds an additional line in the output showing the column totals.
- **-V, – -version :** It displays version info and exit.

2) ***vmstat*** command: **vmstat** command gives the information about processes, memory, paging, block IO, disk and CPU scheduling. All these functionalities make the command ***vmstat*** *which is* also known as *virtual memory statistic reporter*.

Syntax: ***$ vmstat [option]***

-a Active: It displays active and inactive memory of the system running.
-s: It is used to display a table of various event counters and memory statistics
-d: This option is used to display all disk statistics.
top Command
We have already discussed about top command.

3) Commands for disk space usage: Monitoring disk usage and storage space is important to maintain the efficiency of Linux system. Following are some commands that are used to monitor the disk space usage.

- ***du*** command: ***du*** i.e., disk usage, is used to estimate file space usage. The ***du*** command can be used to track the files and directories which are consuming excessive amount of space on hard disk drive.
- ***df*** command: ***df*** command is used to display the space consumed by a particular file system on LINUX system or how much space is available on a particular file system.



CHECK YOUR PROGRESS

Q.2: State whether the following statements are true (T) or false (F):

i) **userdel** command is used to create a new user account.

ii) **passwd** command is used to change the account password.

iii) ***top*** command is used to display the list of open files.

iv) ***du*** command is used to estimate file space usage.



12.9 LET US SUM UP

- Root account is also known as superuser, it has complete and full control of the system.
- To become superuser we may login as superuser or we may execute the ***su*** command from other user account.
- With the help of **exit** command or **Ctrl+D** we may exit from the superuser
- **useradd** command create a new user account and adds an entry to the **/etc/passwd, /etc/shadow, /etc/group** and **/etc/gshadow** files
- The **usermod** command is simple to use with lots of options to make changes to an existing use.
- Superuser can delete an existing user account with **userdel** command.
- **passwd** command is used to change the account password.
- ***vmstat (Virtual Memory Statistics*)** command used to display statistics of virtual memory.
- UNIX provides a general-purpose logging facility called **syslog.**
- **free** command displays the total amount of free space available.
- ***du*** and ***df*** commands are used to monitor disk space usage.



12.10 FURTHER READING

1) Petersen, R. (2007). *Linux: the Complete Reference*. McGraw-Hill, Inc.

2) Siever, E., Weber, A., Figgins, S., Love, R. & Robbins, A. (2005). *Linux in a Nutshell*. O'Reilly Media, Inc.



12.11 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: i) Superuser; ii) exit; iii) useradd; iv) usermod

Ans. to Q. No. 2: i) False; ii) True; iii) False; iv) True



12.12 MODEL QUESTIONS

Q.1: Who is a superuser?

Q.2: What are the different ways to become a superuser?

Q.3: Briefly discuss how to create new user account.

Q.4: What is usermod command? What are the different options that can be used in usermod command?

Q.5: What is userdel command? What are the different options that can be used in userdel command?

Q.6: How can we change the password of a user account?

Q.7: Briefly discuss some system monitoring commands.

Q.8: What is syslog? Explain.

Q.9: What are the different commands used to monitor the memory usage and disk space usage?

*** ***** ***

UNIT 13: NETWORKING IN LINUX

UNIT STRUCTURE

13.1 Learning Objectives

13.2 Introduction

13.3 IP Address

13.3.1 Classes of IP Address

13.4 Netmasks

13.5 Network Address

13.6 Ifconfig

13.7 Ping

13.8 Netstat

13.9 Traceroute

13.10 Telnet

13.11 Etc/Services File

13.11.1 Port Numbers

13.12 Let Us Sum Up

13.13 Further Reading

13.14 Answers to Check Your Progress

13.15 Model Questions

13.1 LEARNING OBJECTIVES

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

- describe the basic concept of IP address
- explain the different classes of IP Address
- explain the need of netmasks and the division between networks and hosts
- describe the idea of network address
- describe the netmasks and gateway configuration with ifconfig
- describe how ifconfig utility is used by network scripts to start, stop, and configure the interface device
- explain use of ping command
- describe different options and parameters of Netstat utility

- describe Telnet and trace route applications
- explain the significance of etc/services files and the use of well known ports.

13.2 INTRODUCTION

In the previous units we have learned how to create users and superusers. We have also learned how to manage user accounts by creating new user accounts, by modifying user accounts or by removing an existing account. This unit gives an introduction of IP addresses and its classes. You will be able to understand the use of netmask in network address. A brief idea of the ranges of the IP address classes are also given in this unit and you will be able to understand the netmask configuration under classful addresses and classless addresses. You will also learn about the use of ifconfig to configure and how to reconfigure any interface device. Besides the unit also explains the utilities of Ping, Traceroute, Telnet and Netstat and will help you to understand the use of different well known ports. In the next unit we will explore the working methodology and configurations of different protocols like NFS, NIS, DNS, FTP, DHCP etc.

13.3 IP ADDRESS

Every network device on the Internet has a unique IP address assigned to it. These IP addresses come in a very distinct format. IP addresses are represented in format known as dotted decimal notation. For example: 192.168.2.24. The IP addresses are 32 bits long. 8 bits are used to form each of the dotted decimal numbers, the numbers are referred to as octets.

13.3.1 Classes of IP Address

IP addresses are classified into 5 categories namely- class A, B, C, D and E:

The class A addressing has up to 128 networks and each having 16 million hosts. The range of host addresses of class A addressing is 1.0.0.0 to 127.255.255.255.

The class B addressing has up to 16384 networks and each having 64000 hosts. The range of host addresses of class B addressing is 128.0.0.0 to 191.255.255.255.

The class C addressing has up to 2 million networks and each having 256 hosts. The range of host addresses of class C addressing is 192.0.0.0 to 223.255.255.255.

The Class D addressing is used as multicast address. The range of host addresses of class D addressing is 224.0.0.0 to 239.255.255.255.

The Class E addressing is reserved for future use.

The lowest IP address is 0.0.0.0 and highest IP address is 255.255.255.255. During booting the IP address 0.0.0.0 is used by the host. The IP address with 0 as network number refers to the current network.

13.4 NETMASK

The netmask defines the division between network and hosts. The netmask is a binary number whereby all numbers to the left are ones, and to the right all zeroes. Under the classful system, netmasks would be one of 255.255.255.0, 255.255.0.0, or 255.0.0.0. It is possible to have a netmask of 255.255.255.255, but this is a very special case and used for point to point routing only. Under classless addressing, the netmask will always start with 255, but will change depending on how many ones until the first zero. The following table 13.1 shows the various values possible for any of the three subsequent octets in binary, decimal and hexadecimal.

Table 13.1: Various values possible for any of the three subsequent octets in binary, decimal and hexadecimal

Binary	Decimal	Hexadecimal
00000000	0	00
10000000	128	80
11000000	192	c0
11100000	224	e0

11110000	240	f0
11111000	248	f8
11111100	252	fc
11111110	254	fe
11111111	255	ff

A netmask, then, will consist of anywhere from 8-32 ones (except 31) which defines the network portion of the netmasks. Using the classful addresses, the following netmask provide us with the appropriate VSLM(variable length Subnet Mask) as follows:

255.255.255.0 = / 24

255.255.0.0 = / 16

255.0.0.0 = / 8

255.255.255.255 = / 32

Using this arrangement, any number can be substituted from 8- 32 (except 31) as the variable length subnet mask. The following table 13.2 shows the host availability.

Table 13.2: Host availability

Binary Netmask	Decimal Netmask	VSLM	Number of Hosts
11111111111111111111111111111000	255.255.255.248	/ 29	6
11111111111111111111111111110000	255.255.255.240	/ 28	14
11111111111111111111111111100000	255.255.255.224	/ 27	30
11111111111111111111111111000000	255.255.255.192	/ 26	62
11111111111111111111111110000000	255.255.255.128	/ 25	126
11111111111111111111111100000000	255.255.255.0	/ 24	254
11111111111111111111111000000000	255.255.254.0	/ 23	510
11111111111111111111110000000000	255.255.252.0	/ 22	1022
11111111111111111111100000000000	255.255.248.0	/ 21	2046



CHECK YOUR PROGRESS

Q.1: What is an IP address?

.....

.....

Q.2: How many classes of IP address are available?

.....

Q.3: What is the use of netmask?

.....

Q.4: The range of host addresses of class D addressing is .....

13.5 NETWORK ADDRESS

Network address is the network portion of an IP address. For a Class A network, it is the first byte, or segment of the IP address; for a class B network it is the first two bytes; for a class C network, it is the first three bytes. In each case the balance of the IP address is the host address. Assigned network address is globally unique within the internet.

In the following table:

- **n :** It is a bit used for the network ID.
- **H :** It is a bit used for the host ID.
- **X :** It is a bit without a specified purpose.

```
Class A
   0.  0.  0.  0 = 00000000.00000000.00000000.00000000
 127.255.255.255 = 01111111.11111111.11111111.11111111
                   0nnnnnnn.HHHHHHHH.HHHHHHHH.HHHHHHHH
```

```
Class B
128.  0.  0.  0  = 10000000.00000000.00000000.00000000
191.255.255.255  = 10111111.11111111.11111111.11111111
                   10nnnnnn.nnnnnnnn.HHHHHHHH.HHHHHHHH
```

```
Class C
192.  0.  0.  0 = 11000000.00000000.00000000.00000000
223.255.255.255  = 11011111.11111111.11111111.11111111
                   110nnnnn.nnnnnnnn.nnnnnnnn.HHHHHHHH
```

```
Class D
224.  0.  0.  0 = 11100000.00000000.00000000.00000000
239.255.255.255  = 11101111.11111111.11111111.11111111
                   1110XXXX.XXXXXXXX.XXXXXXXX.XXXXXXXX
```

```
Class E
240.  0.  0.  0 = 11110000.00000000.00000000.00000000
255.255.255.255  = 11111111.11111111.11111111.11111111
                   1111XXXX.XXXXXXXX.XXXXXXXX.XXXXXXXX
```

Leading bits of the Class A, B and C are known as the most significant bits. Class A has a leading bit of 0 and the following 7 bits comprise the network ID, 0-127. Class B has the leading bits 10, which converts to 128-191, and Class C is 110, 192-223.

13.6 IFCONFIG

The **ifconfig** utility is used by network scripts to start, stop, and configure the interface device, and that's where its name comes from: configure network interface. The "ifconfig" command is used for displaying current network configuration information, setting up an ip address, netmask or broadcast address to a network interface, creating an alias for network interface, setting up hardware address and enable or disable network interfaces. This will be either an Ethernet or a token ring card. If called with no parameters, ifconfig will show currently running interfaces. If called with the –a option, all devices that have been configured since the last reboot even if they have been turned off will be shown. It displays the results for eth0, lo, sit0 and tun0.

```
ifconfig -a
```

```
eth0  Link encap:Ethernet  HWaddr 00:0B:CD:1C:18:5A
      inet addr: 172.16.25.126 Bcast: 172.16.25.63 Mask: 255.255.255.224
      inet6 addr: fe80::20b:cdff:fe1c:185a/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:2344927 errors:0 dropped:0 overruns:0 frame:0
      TX packets:2220777 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:293839516 (280.2 MiB)  TX bytes:1043722206 (995.3 MiB)
      Interrupt:185 Memory:f7fe0000-f7ff0000

lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING  MTU:16436  Metric:1
      RX packets:5022927 errors:0 dropped:0 overruns:0 frame:0
      TX packets:5022927 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:2175739488 (2.0 GiB)  TX bytes:2175739488 (2.0 GiB)

sit0  Link encap:IPv6-in-IPv4
      NOARP  MTU:1480  Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

tun0  Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-
      00-00-00-00-00
      inet addr:10.1.1.1  P-t-P:10.1.1.2  Mask:255.255.255.255
      UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500
      Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:100
      RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Using interface name (eth0) as an argument with “ifconfig” command will display details of specific network interface.

```
ifconfig eth0
```

```
eth0  Link encap:Ethernet  HWaddr 00:0B:CD:1C:18:5A
      inet addr:172.16.25.126 Bcast:172.16.25.63 Mask:255.255.255.224
      inet6 addr: fe80::20b:cdff:fe1c:185a/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:2345583 errors:0 dropped:0 overruns:0 frame:0
      TX packets:2221421 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:293912265 (280.2 MiB)  TX bytes:1044100408 (995.7 MiB)
      Interrupt:185 Memory:f7fe0000-f7ff0000
```

The “up” or “ifup” flag with interface name (eth0) activates a network interface, if it is not in active state and allowing it to send and receive information. For example, “ifconfig eth0 up” or “ifup eth0” will activate the eth0 interface.

The “down” or “ifdown” flag with interface name (eth0) deactivates the specified network interface. For example, “ifconfig eth0 down” or “ifdown eth0” command deactivates the eth0 interface, if it is in active state.

Using ifconfig, it can be started, stopped and configured Ethernet card. Almost every entry can be modified, though the card must support changing that parameter. Field that cannot be changed by ifconfig are the Link encapsulation field, which is determined by the type device; the two lines containing card statistics (the TX and RX lines); the number of collisions; and the name of the device itself. All others can be manipulated.

To assign an IP address to a specific interface, the following command with an interface name (eth0) and ip address can be used. For example, “ifconfig eth0 172.16.25.125” will set the IP address to interface eth0.

Using the “ifconfig” command with “netmask” argument and interface name as (eth0) allows to define a netmask to a given interface. For example, “ifconfig eth0 netmask 255.255.255.224” will set the network mask to a given interface eth0.

Using the “broadcast” argument with an interface name will set the broadcast address for the given interface. For example, “ifconfig eth0 broadcast 172.16.25.63” command sets the broadcast address to an interface eth0.

To assign an IP address, Netmask address and Broadcast address all at once using “ifconfig” command with all arguments as given below.

```
ifconfig eth0 172.16.25.125 netmask 255.255.255.224
broadcast 172.16.25.63
```

The “mtu” argument sets the maximum transmission unit to an interface. The MTU allows to set the limit size of packets that are transmitted on an interface. The MTU is able to handle maximum number of octets to an interface in one single transaction. For example, “ifconfig eth0 mtu 1000” will set the maximum transmission unit to given set (i.e. 1000). Not all network interfaces supports MTU settings.

```
ifconfig eth0 mtu 1000
```

The ifconfig utility allows to configure additional network interfaces using alias feature. To add alias network interface of eth0, the following command is used. Please note that alias network address in same sub-net mask. For example, if eth0 network ip address is 172.16.25.125, then alias ip address must be 172.16.25.127.

```
ifconfig eth0:0 172.16.25.127
```

Next, verify the newly created alias network interface address, by using “ifconfig eth0:0” command. If you no longer required an alias network interface or you incorrectly configured it, you can remove it by using the following command.

```
ifconfig eth0:0 down
```

To change the MAC (Media Access Control) address of an eth0 network interface, use the following command with argument “hw ether”. For example, see below.

```
ifconfig eth0 hw ether AA:BB:CC:DD:EE:FF
```



CHECK YOUR PROGRESS

Q.5: How many bits of class A addressing represent network id?

.....

.....

Q.6: What is the use of ifconfig utility?

.....

.....

.....

.....

Q.7: What will be displayed if ifconfig command is called with option –a?

.....

.....

13.7 PING

PING (Packet Internet Groper) command is used to check the network connectivity between host and server/host. It also helps you assess the time it takes to send and receive a response from the network. IP address or the URL is provided as input to this command and sends a data packet to the specified address with the message “PING”.

Ping uses **ICMP(Internet Control Message Protocol)** to send an **ICMP echo message** to the specified host if that host is available then it sends **ICMP reply message**. Ping is generally measured in millisecond. The following command shows the use of ping with the IP address:

```
Ping 192.168.0.2 or ping www.google.com
```

Ctrl+c is used to stop the ping. The number of packets to be sent to the server/host can be defined by using the option –c.

```
ping -c 5 www.google.com
```

The size of the packets to be sent to the server/ host also can be defined by using the option –s.

```
ping -s 40 -c 5 www.google.com
```

By default, ping waits for 1 sec to send next packet but it can be changed by using **- i** option.

```
ping -i 2 www.google.com
```

Now, the ping interval will change to 2 seconds.

To only get the summary about the network use **-q** option. Ex: ping -c 5 -q www.google.com.

To stop pinging after sometime, use **-w** option. E.g.: ping -w 3 www. google.com. This will stop pinging after 3 seconds. To send packets as soon as possible –f option is used. This is used to test network performance. ping -f www. google.com

To record the current time of event by a machine over a network, **ts** option of IP packet is used. Three option are as follows:

1) **tsonly** (timestamp only)
2) **tsandaddr** (timestamp and address)
3) **tsprespec** (timestamp pre-specified for multiple hosts)

```
ping -T tsonly -c 2 127.0.0.1
ping -T tsandaddr -c 2 127.0.0.1
```

To set time to wait for a response, following command is used

```
ping -c 5 -W 3 www.google.com
```

13.8 NETSTAT

The **netstat** utility is a multifaceted utility. **Netstat** command displays various network related information such as network connections, routing tables, interface statistics, masquerade connections, multicast memberships etc. If anything is required from routing information to open connections to servers bound to ports, it is needed to use netstat. The netstat will show what ports are connected to from where, and what ports are ready to accept connections. It will show the state of connections. Executing netstat with no

options will show the state of active network connections. By default, netstat resolves both hostnames and service names. If there is no need of resolving names, but you need to see numbers and then use -n option. The option –a will show all the bound services. Adding the –v switch to netstat with or without the –a will show which network protocols are not supported. The –r option will show routing, and when combined with –e or –ee will provide slightly different parameters. Some states of the connections are mentioned below:

ESTABLISHED: An active connection is made.
SYN_SENT: A new connection is being initiated with a foreign host.
SYN_RECVED: A new connection is being initiated with a foreign host.
FIN_WAIT1: Socket closed, connection shutting down.
FIN_WAIT2: Connection closed, waiting for shutdown from remote host.
TIME_WAIT: Connection closed, socket is waiting to handle packets still onn the network.
CLOSED: socket is not in use.
CLOSE_WAIT: Remote host is shutdown, waiting for socket to close.
LAST_ACK: The remote host is shut down, socket is closed, waiting for remote acknowledgement
LISTEN: Socket is bound by a server and waiting for a connection.
CLOSING: Both sockets are shut down, but packets remain to be sent.
UNKNOWN: You should never see this.

Following are some examples of commands used in netstat:

 # netstat -at : To list all tcp ports.
 # netstat -au : To list all udp ports.
 # netstat -l : To list only the listening ports.
 # netstat -lt : To list only the listening tcp ports.
 # netstat -lu : To list only the listening udp ports.
 # netstat -s : To list the statistics for all ports.
 # netstat -st(TCP) : To list the statistics for TCP ports.
 # netstat -su(UDP) : List the statistics for UDP ports.
 # netstat -c : To print the netstat information continuously.

 # netstat —verbose : To get the non-supportive address families in the system.

 # netstat -r : To get the kernel routing information.

 # netstat -ap | grep ssh : To get the port on which a program is running.

 # netstat -an | grep ':80' : To get the process which is using the given port.

 # netstat -i : To get the list of network interfaces.

 # netstat -ie : To display extended information on the interfaces

13.9 TRACEROUTE

The traceroute command prints the route that a packet takes to reach the host. It is useful to know about the route and about all the hops that a packet takes. If it is possible to ping hosts on the local network, but are unable to determine whether a good host have been selected on the Internet or not, then it is required to switch to traceroute. The traceroute utility will show the path through which packets reach a destination. The good thing about the traceroute is that even if the remote host is down, it is possible to trace the systems through which the packets will travel and where the network finally stops. If a remote system is chosen to see where the packets are trying to go, use the following command taking an IP address:

```
traceroute [options]  host_Address [pathlength]
```

If the packets stop at the gateway then there is a problem with gateway. If the packet travels beyond the gateway, then the part of the network chosen is working and need to wait for other administrators to work on the rest. The traceroute utility with the option –n tells traceroute to use the numeric destination of hosts rather than trying to resolve the names.

-4 Option: Use ip version 4 i.e. use IPv4

```
$ traceroute -4 10 google.com
```

-6 Option: Use ip version 6 i.e. use IPv6

```
$ traceroute -6 10 google.com
```

-F Option: Do not fragment packet.

```
$ traceroute -F google.com
```

-f first_ttl Option: Start from the first_ttl hop (instead from 1)

```
$ traceroute -f 10 google.com
```

-g gate Option: Route the packet through gate.

```
$ traceroute -g 192.168.43.45 google.com
```

-m max_ttl Option: Set the max number of hops for the packet to reach the destination. Default value is 30.

```
$traceroute  -m 5 google.com
```

-n Option: Do not resolve IP addresses to their domain names.

```
$traceroute -n google.com
```

-p port Option: Set the destination port to use. Default is 33434.

```
$traceroute  -p 20292 google.com
```

packetlen Option: The full packet length. Default len is 60 byte packets.

```
$traceroute  google.com 100
```

–help: Display help messages and exit.

```
$traceroute –help
```

13.10 TELNET

The telnet utility implements the TELNET protocol to connect to a remote computer over a network. The telnet utility is not secure. **Telnet** is the joint abbreviation of **Telecommunications and Networks.** It sends username and password over the network in cleartext, which is not secure

practice. There are no authentication policies & data encryption techniques used in telnet causing huge security threat. That is why telnet is no longer used for accessing network devices and servers over public network. When remote system name is not specified, telnet works interactively and prompts to enter one of the commands mentioned below:-

-a : Initiates automatic login

-K: Prevents automatic login.

-l *username:* It attempts an automatic login on the remote system using *username*. If the remote system understands how to handle automatic login with telnet, it prompts for a password.

-e *c:* It changes the escape character from control-] to character *c.*

After telnet connects to remote system, telnet can be put into command mode by typing the escape character (usually CONTROL-]). The remote system reports the escape character it recognizes. To leave the command mode, we need to type RETURN on a line by itself. Few commands of telnet are mentioned below:

? : (help) Displays a list of commands recognized by the telnet utility on the local system.

close : Closes the connection to the remote system.

logout : Logs out of the remote system , it is similar to close.

quit : Quits the telnet session.

z : Suspends the telnet session. When a session is suspended, then it returns to the login shell on the local system. To resume the suspended telnet session, type fg at a shell prompt.

When a remote system is connected, telnet presents a textual login: prompt. Since telnet is designed to work with non UNIX and non Linux systems, it does not assume that the remote username is the same as the local username. In some cases, telnet requires no login credentials. In addition, telnet allows to configure special parameters, such as how RETURNs or interrupts are processed.

By default telnet connects to port 23, which is used for remote logins. If telnet is used to connect to port 25 on a host, then it can interact with SMTP. In addition, port 110 connects to the POP protocol, port 80 connects with a www server, and port 143 connects to IMAP. All of these are ASCII protocols.

13.11 /etc/services FILE

The **/etc/services** file is a plain ASCII file providing a mapping between human-friendly textual names for internet services, and their underlying assigned port numbers and protocol types. Every networking program should look into this file to get the port number (and protocol) for its service. The C library routines getservent(3), getservbyname(3), getservbyport(3), setservent(3), and endservent(3) support querying this file from programs. Port numbers are assigned by the IANA (Internet Assigned Numbers Authority), and their current policy is to assign both TCP and UDP protocols when assigning a port number. Therefore, most entries will have two entries, even for TCP-only services. Port numbers below 1024 are called as “low numbered” ports which can be bound to only the root. The presence of an entry for a service in the **services** file does not necessarily mean that the service is currently running on the machine. All networking services are started by **inetd**(8). News (NNTP) and mail (SMTP) servers are often initialized from the system boot scripts.

The /etc/services file is used by applications to translate human-readable service names into port numbers when connecting to a machine across a network. The file will typically include the service name, port/protocol, any aliases, and comments. A protocol being listed in the /etc/services file does not mean that the port is blocked or unblocked by a firewall or a security mechanism. The default entries should not be modified in /etc/services file but new entries can be added as per the requirement. This file is a good resource to see which services use which port numbers and the protocols that are specified to work.

The location of the **services** file is defined by **_PATH_SERVICES** in *<netdb.h>*. This is usually set to */etc/services*.

Syntax is of the form:

```
service-name   port / protocol   [aliases ...]
```

service-name is the friendly name the service is known by. It is case sensitive. Often, the client program is named after the *service-name*.

port is the port number (in decimal) to use for this service.

protocol is the type of protocol to be used. This field should match an entry in the protocols(5) file. Typical values include **tcp** and **udp**.

aliases is an optional space or tab separated list of other names for this service. Again, the names are case sensitive.

13.11.1 Ports

When dealing with information passing from one computer system to another, information passes between a server application and a client application. Each server program will have default port that it binds to (listen on) for request coming from a client. The client is built to send request to the default server port. The Internet Assigned Numbers Authority (IANA) designates ports for certain services to bind to. Some of these include the following ports and services.

Port	Service
21	FTP
23	TELNET
25	SMTP
80	HTTP(www)
110	pop3
143	IMAP

A complete listing is available in /etc/services file. The service file format is as follows:

```
service port_number/protocol # optional comments
```

Sometimes, new port may be added to the service file. For example, mysql port 3306/tcp may be added. The port number 3306

brings up another point. Ports 0-1023 are reserved. Only root (UID 0) can bind to these lower ports, known as privileged ports. The remaining ports 1024-65535 anyone (including root) can bind. These ports are used as outgoing ports by the client software. They work as incoming ports from the server as well.



CHECK YOUR PROGRESS

Q.8: What is the port number for ftp service?

.....

Q.9: What is the use of TELNET?

.....

.....

Q.10: When SYN_SENT status is displayed?

.....

Q.11: What is the utility of Traceroute?

.....

.....

.....

.....



13.12 LET US SUM UP

- IP Address is a unique address which is assigned to any network device connected to internet.
- The netmask defines the division between network and hosts.
- The “ifconfig” command is used for displaying current network configuration information, setting up an ip address, netmask or broadcast address to a network interface, creating an alias for network interface, setting up hardware address and enable or disable network interfaces.
- PING (Packet Internet Groper) command is used to check the network connectivity between host and server/host.

- Netstat command displays various network related information such as network connections, routing tables, interface statistics, masquerade connections, multicast memberships etc.
- The traceroute utility shows the path through which packet reaches a destination.
- **Telnet** is the joint abbreviation of **Telecommunications and Networks.** It sends username and password over the network in cleartext, which is not secure practice.
- The etc/services file is a plain ASCII file providing a mapping between human-friendly textual names for internet services, and their underlying assigned port numbers and protocol types.



13.13 FURTHER READING

1) Bandel, D., & Napier, R. (2000). *Special Edition Using Linux*. Que Corp.
2) Sobell, M. G. (2009). *A Practical Guide to Linux Commands, Editors, and Shell Programming*. Pearson Education.



13.14 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: IP address is a unique address assigned to every network device connected to Internet.

Ans. to Q. No. 2: There are five classes of IP addresses.

Ans. to Q. No. 3: The netmask defines the division between network and hosts.

Ans. to Q. No. 4: The range of host addresses of class D addressing is 224.0.0.0 to 239.255.255.255.

Ans. to Q. No. 5: Class A has a leading bit of 0 and the following 7 bits comprise the network ID, 0-127

Ans. to Q. No. 6: The “ifconfig” command is used for displaying current network configuration information, setting up an ip address, netmask

or broadcast address to a network interface, creating an alias for network interface, setting up hardware address and enable or disable network interfaces

Ans. to Q. No. 7: This option is used to display all the interfaces available, even if they are down.

Ans. to Q. No. 8: The port number for ftp service is 21.

Ans. to Q. No. 9: The telnet utility implements the TELNET protocol to connect to a remote computer over a network.

Ans. to Q. No. 10: When a new connection is being initiated with a foreign host.

Ans. to Q. No. 11: Traceroute utility will show the path through which packets reaches a destination. The good thing about the traceroute is that even if the remote host is down, it is possible to trace the systems through which the packets will travel and where the network finally stops.



13.15 MODEL QUESTIONS

Q.1: What is an IP address? What are its classes and ranges?

Q.2: What is the use of Netmasks?

Q.3: Explain the significance of network address to use internet.

Q.4: What is ifconfig?

Q.5: Discuss the different options of ifconfig utility.

Q.6: What is the advantage of traceroute utility?

Q.7: What is the main drawback of TELNET application?

Q.8: Discuss the different state of connection of Netstat command.

Q.9: Discuss the different port numbers assigned for specific services.

Q.10: Explain the different options of ping command.

*** ***** ***

UNIT 14: NETWORK PROTOCOLS

UNIT STRUCTURE

14.1 Learning Objectives

14.2 Introduction

14.3 Network Protocols

14.4 Network File System (NFS)

- 14.4.1 Configuration of NFS Server
- 14.4.2 Configuration of NFS Client

14.5 Network Information System (NIS)

- 14.5.1 Component of NIS
- 14.5.2 Steps to Configure NIS

14.6 Domain Name System (DNS)

14.7 File Transfer Protocol (FTP)

14.8 Squid Proxy

14.9 DHCP server

14.10 Iptables

14.11 Let Us Sum Up

14.12 Further Reading

14.13 Answer to Check Your Progress

14.14 Model Questions

14.1 LEARNING OBJECTIVES

After going through this unit you will be able to:

- explain the concepts and use of network protocols
- describe the functions and configurations of NFS
- explain the use of NIS in Linux
- describe the necessity of DNS
- transfer files through FTP in networking
- describe the working and benefits of DHCP server
- discuss the importance of iptables.

14.2 INTRODUCTION

We are already familiar with the functions of different layers of OSI and TCP/IP models. In the previous units, we have studied the networking process in Linux system and different commands to access them and have gathered knowledge about the different services and their port numbers. We have also learned about the netmask configuration under classful addresses and classless addresses and how to use ifconfig to configure and how to reconfigure any interface device. In this unit we will learn about what a network protocol is. We will study the working methodology and configurations of different protocols viz. NFS, NIS, DNS, FTP, DHCP etc. We will discuss the use of iptables and firewalls. In the next unit we will learn about different security issues in network like cyber attacks and the solutions to make us safe from those attacks like TCP-IP and ssh.

14.3 NETWORK PROTOCOL

A protocol is defined as a set of rules. A network protocol is a set of rules for communicating between two or more network devices or nodes. It includes the detail mechanisms of how to make connections between two networking devices and establish the special format to create the transmission and reception of packages between them. These protocols are used for secure end to end process communication timely. These protocols must be installed and configured by the sender and the receiver to perform network communication. There are various types of network protocols which are commonly used such as internet protocols, routing protocols etc.

14.4 NETWORK FILE SYSTEM (NFS)

It is a distributed file system protocol that allows a user to access remote disk and file system locally. Through NFS, a client computer can access all the files over a computer network. It is based on the **ONC RPC** system. **ONC RPC** stands for Open Network Computing Remote Procedure Call. NFS was initially developed in 1984 by **Sun Microsytems.**

NFS provides accessing of files through an interface known as **Virtual File System (VFS)**. Users can access and perform different operations on the shared file as they can if the file is stored in the local system. NFS is created based on the client-server architecture. In NFS, client computers access the remote files and directories shared by the server computer. A server computer manages and takes care of the authentication issues of clients system.

The main advantages of NFS are listed below:

- NFS provides local access to remote file system.
- Users can access their data irrespective of any physical location.
- The workload of administrator can be reduced by using a centrally managed server through NFS.
- Users can access their files by logging into any of the distributed servers.
- Centralized storage solution can be configured using NFS.

Some of limitations of NFS are described as follows:

- Lack of security is the major disadvantage of NFS. Since NFS is based on **Remote Procedure Call (RPC)**, so NFS should only be used on a trusted network.
- For NFS, the performance of network may be slow at the time of heavy network traffic.

14.4.1 Configuration of NFS Server

Different steps to configure a NFS server are listed below:

Step 1: First, we need to install the required packages or software. For this, we need to access the system as super user. The nfs-common package is available for both the client and the server. We can install this using the following command:

```
# apt –get install nfs-common
```

This command installs all the required files to process the NFS. Depending on the Linux distribution, the installation command may vary. If required, nfs-kernel-software also needs to be installed separately.

Step 2: Suppose, we want to share a specific folder named as '**shared_folder1'**. For this we need to use:

```
# mkdir shared_folder1
```

Step 3: We can create some files inside the directory using touch command:

```
shared_folder1#  touch test1  abc.txt
pqr.txt
```

Step 4: We need to edit the /etc/exports file using *gedit* command.

```
# gedit  /etc/exports
```

Inside the file, the following lines are required to add:

```
/home/user1/shared_folder1  *(rw,  sync,
no_root_squash, no_subtree_check)
```

Instead of this, different options can be used. Explanations of the above options are stated below:

rw: This allows the clients to read and write the files of the shared directory.

sync: It is used to confirm the shared directory after modification is done.

no_root_squash: It helps to connect root user to the shared directory. If this option is used, remote root users can change on any file of the shared directory or file system.

no_subtree_check: Scanning of shared directory is restricted using this option. Usually NFS scans every shared directory. This option is used to increase the reliability but it reduces the security.

Step 5: We may require starting the *rpcbind* service:

```
#service rpcbind start
```

Step 6: Next, we need to start (restart) the nfs-common and kernel server:

```
# /etc/init.d/nfs-common restart
# /etc/init.d/nfs-kernel-server restart
```

Step 7: We type the following command to view the where the file system is exported.

```
# exportfs –v
```

Step 8: We type the following command to display all the files that are shared:

```
# showmount –e
```

14.4.2 Configuration of NFS Client

After configuration of NFS server, we need to configure the client. We need to mount the shared directory of the NFS server in the /mnt directory of client. The steps to configure NFS in client side are as follows:

Step 1: Install the required software like *nfs-common* using apt-get or other installation command after becoming super user.

Step 2: Create a directory inside */mnt* directory.

```
# mkdir  /mnt/nfs_client1
```

Step 3: Type the following command to mount the shared directory of NFS server to the */mnt/nfs_client1* directory.

```
# mount 10.10.127.251:/home/user1/
shared_folder1   /mnt/nfs_client1
```

Step 4: To get confirmation about shared directory is mounted or not, type the following command:

```
# df –h
```

This command displays the list of mounted directories.

14.5 NETWORK INFORMATION SYSTEM (NIS)

Network Information System or NIS is a client– server protocol based on **RPC**. NIS is also known as **Yellow pages (YP)** which was initially developed by Sun Microsystems. It allows a group of computers to share a common set of files and directories within themselves. An NIS domain name is used for a group of systems present in a subnet that are using the same NIS map.

In NIS, each host whether it is client or server has knowledge about the entire system. A user at any machine can access any files or application of any host in the network using the single user identification and password.

14.5.1 Components of NIS

NIS is composed of two parts viz. NIS server and NIS client. They are grouped together in a domain. Each domain has set of characteristics.

NIS Server: NIS server is a host computer which provides configuration services to the other hosts or clients present in the network. The server runs the *ypserv* daemon. It is used for processing the requests from clients for information contained in maps. NIS servers are of two types:

- **Master Servers:** A master server in NIS is a single host in a specific NIS domain. It maintains the maps. This server runs *ypupdated* daemon. From this slave servers update a copy of their maps. Another daemon called *yppasswdd* also runs by the master server. It processes the requests to change the user's passwords.
- **Slave Servers:** NIS slave servers maintain a copy of NIS master servers data files. If the number of hosts in the network is less then each host can access the master server directly. But if the number of hosts in the network is more, server may become overloaded. So, the slave servers are needed as a backup of master servers. If for some reason master server fails, the slave server takes the responsibility immediately.

NIS slave servers usually act as a mediator between the master server and the clients. NIS slave servers maintain copies of the NIS master's data files. Slave servers also help to balance the load of the master server. A slave server can response on the same way the master server can respond since slave server keeps exact replica of master server files. The changes are directly mapped in the master server rather than slave servers. So, slave servers can help to reduce the workload of master server. In every NIS, there should be at least one slave server. The balance of number of slave servers in NIS domain should be maintained.

NIS Clients: NIS clients run a daemon called *ypbind* which allows a client process to get information from the server. For accessing the system information in a map, the client makes a Remote Procedure Call (RPC) to the server. The server searches it in stored NIS database and provides information to the client.

NIS Domains: An NIS domain is a group of hosts which are logically connected together. All the hosts in a group must have the same domain name. In each NIS domain, there is a NIS master server. But if the number of hosts increases, the performance may decrease. We cannot add more than one master server for one NIS domain. So, for this slave servers are used as a backup of master server.

NIS Maps: NIS maps or databases provide information such as user names, their passwords etc. The information in a database is stored in DBM (DataBase Management) format.

There are four different types of Network Information System can be configured:

- An NIS with one single domain with a master server, no slave server and one or more clients.
- An NIS with a single domain with a master server, one or more slave severs and different number of clients.
- An NIS with multiple domains each with its own master server and one or more clients where there is not any lave server.
- An NIS with multiple domain each with its own master server and one or more clients along with one or more slave servers.

14.5.2 Steps to Configure NIS

The simplest NIS configuration consists of one single NIS server and one or more clients. We will study only the main steps required for NIS installation and configuration in both server and clients. For configuring a NIS server following steps have to be carried out:

Step 1: Install the required software or packages.

Step 2: Set the NIS domain name

Step 3: Initialize the NIS map

Step 4: Start the NIS password daemon

Step 5: Starting the NIS transfer daemon if we use the slave servers.

Step 6: Modify the starter process to start the NIS daemon at the time of system reboots.

If we have slave servers, we need to configure the slave servers also using the above steps. To configure NIS client, following steps need to be carried out:

Step 1: Install the required software

Step 2: Set the NIS server domain name

Step 3: Configure and start the NIS client daemon

Step 4: Configure the client start up files to user NIS.

Step 5: Reboot the client.

These are the steps to follow to configure Network Information System.

14.6 DOMAIN NAME SYSTEM (DNS)

Domain Name System or DNS is a hierarchy of namespace used for mapping on domain name and IP address. It is a distributed database for storing and managing the entire domain names with their corresponding IP addresses. A DNS name server is a server that stores the DNS records for a domain and provides answer to the queries against its database. The functions of Domain Name System are stated below:

- DNS maps the website address entered by a user to its IP address.
- DNS maintains the domain name hierarchy and translates services between the address space and it.

- It works a resolver for query in the database.
- It improves routing and sender authentication procedure.
- DNS is used by name servers to exchange information.

DNS clients are also known as **DNS resolver**. Whenever a query is sent, DNS resolver will initiate it. DNS resolver checks whether the host name is available in local cache. If it is not, it will pass to the DNS name servers. DNS server responds the queries from their own domains or from different domains that come from outside. There are three different types of queries

- **Recursive Query:** In this type of query, the DNS resolver asks to single DNS server. That server may send the query to another server recursively on behalf of the resolver.
- **Iterative Query:** In this type of query, resolver asks to one or more severs. Each server refers to another server in the chain until the request is fully resolved by the current server.
- **Non- Recursive Query:** In this type of query, resolver checks whether it is present in the cache or not. If not, it asks to a DNS server which is authoritative.

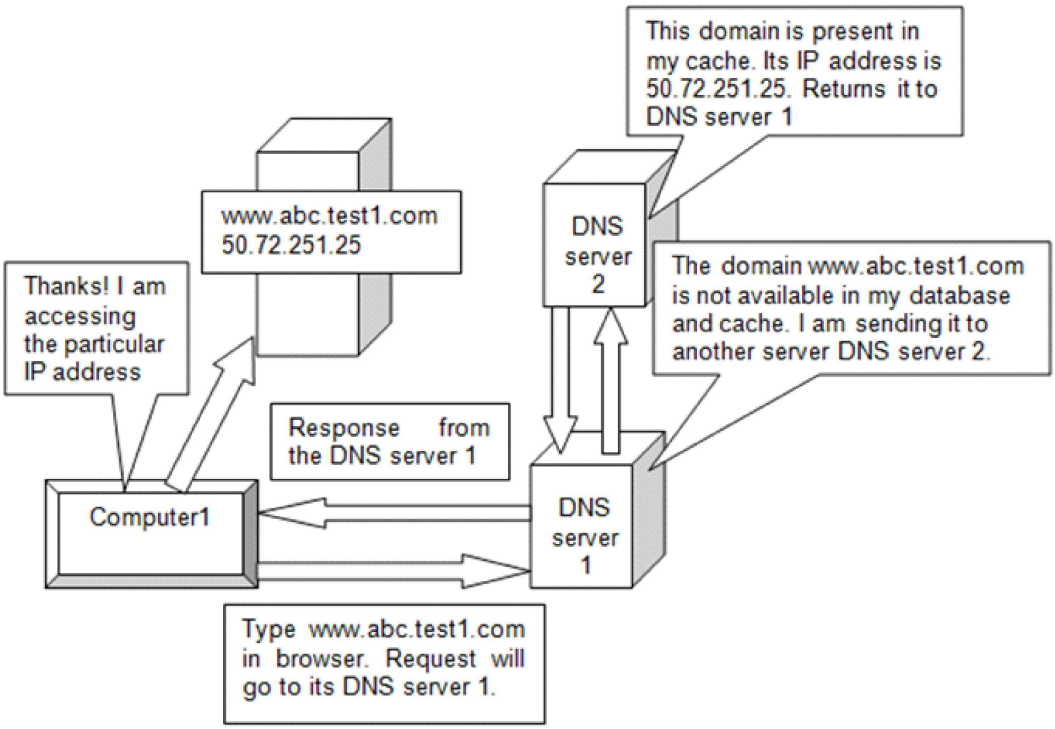


Figure 14.1: Working of Domain Name System

The working of DNS server is as in figure 14.1. First, let the user type the URL www.abc.test1.com (say) in the browser of a computer. The browser will send it to its own domain server for asking about its IP address. The domain server named DNS server 1 (in this case) will receive the query and search it in its cache and database. If it is present, it will send the IP address to the client. If it is not, it will send to the other DNS server named DNS server 2 (as in figure 14.1) based on Top Level domain.

After receiving the request from DNS server 1, DNS server 2 will search it in its cache, and then in the database. If the mapping is present there, it will send back to DNS server1 and DNS server1 will send it to the respective client. As in the figure 14.1, DNS server 2 finds the mapping in its cache and returns the IP address 50.72.251.25 (say) to DNS server 1. After receiving the IP address, the client will access that domain through the browser. If the website requested by the client is not available, page not found or error occurs the message will be sent. This is how the Domain Name System works in the internet.

Some of the popular public DNS servers are viz. Google public DNS server, OpenDNS, Comodo Secure DNS, DNS.Watch etc.



CHECK YOUR PROGRESS

Q.1: NFS stands for .....

a) Network File Server b) Network File System

c) Netware File System d) Netflow File Server

Q.2: The client side of DNS client is called as .....

a) DNS updater b) DNS handler

c) DNS seeker d) DNS resolver

Q.3: Which of the following daemons is run by NIS server to process the requests from clients for information contained in map?

a) ypserv b) ypupdated

c) yppasswdd d) yppush

Q.4: A DNS database contains which of the following?

a) Hostname to address records b) Name server record

c) Hostname aliases d) All of the above

Q.5: "no_root_squash" option in /etc/export file means:

a) Do not connect root user to the shared directory

b) Restrict scanning of shared directory

c) Connect root user to the shared directory

d) Allow scanning of shared directory

Q.6: Which of the following cannot be true?

a) Multiple hostnames may correspond to a single IP address.

b) A single hostname may correspond to many IP addresses

c) A single hostname may correspond to a single IP address

d) None of the above

14.7 FILE TRANSFER PROTOCOL (FTP)

File Transfer Protocol or FTP is commonly used for transferring files between two communicating devices in a computer network. It uses TCP/IP protocol. FTP is a client – server architecture based protocol where the client and the server can transfer files between them. For this, the username and password are required, but if the server is configured to allow the transfer, it can happen anonymously also.

FTP is mainly used to download any files from the server and for uploading files to the server. We can establish FTP connection between any two computers where one is client and the other computer acts as server. The client machine is known as local host and the server machine is known as remote host. Both can exchange files between themselves. FTP works exactly like **HyperText Transfer Protocol (HTTP)**. So, we can connect to FTP address through any web browser. Though FTP can be accessed anonymously without user name and password, but security is less there. It is always advisable to a secure transmission with protected user name, password, and encrypted content. For secure FTP transmission, FTP is replaced with **SSH File Transfer Protocol (SFTP).**

In the next section, we will configure Secured File transfer Protocol and will see how transfer files through SFTP in Linux:

Step 1: First, required software need to be installed in both client and server. After that, we will access the remote working directory of a server computer whose name is **server1** (say) from a clients computer **user1** (say)**.**

Step 2: Need to become a super user or root user from client's computer using su or sudo command.

Step 3: Type the following command to connect with the server using **sftp.**

```
/home/user1# sftp server1@10.10.127.251
```

It will ask for password. After authenticated, the terminal looks like:

```
sftp>
```

Step 4: Type **lpwd to** view local working directory.

```
sftp>    lpwd
```

Step 5: Type **pwd** to see remote working directory

```
sftp> pwd
```

Step 6: For getting any file (say **xyz.jpg**) stored in remote working directory, type:

```
sftp>     get ./xyz.jpg
```

This command will download this file to the local working directory successfully.

Step 7: For moving or uploading a file (say **lmn.txt**) stored in local working directory to the server, type:

```
sftp>     put ./lmn.txt
```

This command will upload the file to the working directory of the server. We can view the file using **ls** command. We can upload

files from any directory by changing the directory using **cd** command or by putting the path of the file name.

Step 8: After transmission, for close the connection type:

```
sftp>      bye
```

Get command is used to download the file where **put** command is used to upload the file in File Transfer Protocol.

14.8 SQUID PROXY

Squid is a web proxy caching server application. It provides services to HTTP, HTTPS, FTP and other common network protocols. Squid was initially designed for UNIX like system. It is free and open source software.

A proxy server allows all the client computers to connect to any network resources indirectly. If a client requests for getting some network services like any website or particular link, the request will first be sent to the proxy server (in case, proxy is installed). If it is available in cache, proxy will immediately return it to the client. If not, it will pass it to the relevant server.

A squid proxy server is commonly installed on a separate server not in the web server with the original files. After installation of squid server, the web server is configured for using the squid server as a proxy. First, squid will act as an intermediary and help to pass the client's request on to the server. Along with this, it will keep a replica of the requested object. Later, if the same client or any other client sends request for the same object and it is present in squid's cache, it will immediately process it. A squid proxy server has various advantages. These are listed below.

- Multiple devices can share internet from one connection through proxy server.
- It increases the speed of web server by caching repeated request. The entire request will reach to the proxy first, if it is present there, response will come directly. So, it helps to increase the speed.
- Since proxy server acts as an intermediary, so it can hide the addresses of the client computer. It is beneficial in case of security reasons.

- Squid proxy is used for SSL requests and caching of Domain Name Server lookups.
- For security reasons, restriction may be implemented i.e. we can block many websites in the network if needed.
- Scanning of content like data packet header can be performed.

14.9 DHCP SERVER

DHCP stands for **Dynamic Host Control Protocol.** It is used to assign IP address to any computer on a network dynamically. Along with IP address, it also assigns other network configuration such as subnet mask, directed Broadcast ID etc. It reduces the work of network administrator by assigning the required network parameters automatically from Internet Service Provider (ISP) to a host in the network. If DHCP server is not available, any computer or other devices need to configure IP address and other parameters manually.

DHCP operates on the basis of a client – server model. When a device is connected to a network, then the installed DHCP application will send a request to the DHCP server. DHCP server will provide the service to the client. DHCP server manages a list of IP addresses and other network information of different clients. DHCP will respond to the client with required information based on the assignment policy set by the particular administrator. Generally, a DHCP client asks for information immediately after booting or before the expiration of previous information, as the information changes periodically. So, each time DHCP server may assign new information based on the availability and assignment policies.

DHCP server stores various information in its database like valid TCP/ IP configuration parameter, IP addresses, reserved IP addresses with particular clients, the time period of a valid lease. The main advantage of using DHCP server is that it reduces the amount of time needed as well as reduces the work of network administrator for configure the appropriate IP address, subnet mask, port number, gateway or any other specific information. DHCP provides a reliable IP address configuration method.



CHECK YOUR PROGRESS

Q.7: FTP is based on ..... model.

a) P2P b) Client – Server

c) Top- Down d) Bottom – Up

Q.8: IP assigned for a client by DHCP server is:

a) for a limited period b) for unlimited period

c) not time dependent d) none of the above

Q.9: Which of the following is true respective of DHCP server?

a) maintains a database of available IP addresses

b) maintains the information about client configuration parameters

c) grants a IP address when receives a request from a client

d) all of the above

Q.10: FTP is used for:

a) Transfer file from server to client

b) Transfer file from client to server

c) File can be accessed remotely, not transferred

d) Both (a) and (b)

14.10 IPTABLES

Iptables is a utility program that uses a set of tables containing set of rules and chains. It is a utility program which allows the system administrator to configure the tables provided by Linux kernel firewall. Iptables can be said to be the built-in firewall in Linux. It is a default firewall installed in many Linux distributions like Red Hat, CentOS etc. Iptables operate in the third layer i.e. network layer of OSI model.

There are different modules of iptables available for different protocol. E.g. iptables for IPv4, ip6tables are for IPv6 protocol, ebtables for Ethernet frames, arptables for ARP etc.

Three main types of iptables present are viz. Filter, NAT, Mangle. Number of chains are present in the iptables viz. input chain, output chain,

forward chain, RH-Firewall-1 input etc. Some important commands related to iptables are discussed here:

- To check the status of Iptables:

```
# iptables -L -n -v
```

where,

-L this is used to list the rules in a chain or all chain

-n this is used to get the numeric output of address and ports

-v this I used for verbose mode

- To block an address (say xxx.xxx.xxx.xxx) in Iptables firewall:

```
# iptables -A INPUT -p tcp -s xxx.xxx.xxx.xxx -j DROP
```

- To save the changes in iptables:

```
# iptables-save
```



CHECK YOUR PROGRESS

Q.11: A proxy firewall filter works in which layer of OSI model?

a) Data Link Layer b) Network Layer

c) Transport Layer d) Application Layer

Q.12: Iptables operate in which layer of OSI model?

a) Data Link Layer b) Network Layer

c) Transport Layer d) Application Layer

Q.13: For which of the following reason, a proxy server is used?

a) To provide security against unauthorised users

b) To process clients request for database access

c) To process clients request for web pages

d) None of the above

Q.14: Firewall needs to be ..... so that it can grow with the network which it protects.

a) Robust b) Fast

c) Expansive d) Scalable



14.11 LET US SUM UP

- Network protocol is a set of rules and convention for communicating between two devices over a network.
- NFS is distributed file system based on client server architecture used for access the shared files and directories by the server to the clients over a network.
- NIC is also a client-server protocol based on RPN, initially known as Yellow Pages (YP).
- NIS allows a group of computers both the server and clients to access the shared files and directories over a network.
- DNS is a distributed database for storing and managing the entire domain names with their corresponding IP addresses.
- FTP is used to transfer file between two communicating devices over a network.
- Server and clients can exchange files over a network through FTP.
- Squid proxy acts as an intermediary and helps to pass the client's request on to the server.
- DHCP server is used to assign IP address and other detail to a device on a network dynamically.
- Iptables is a utility program that uses a set tables containing set of rules and chains.
- Iptables are known as default firewall.



14.12 FURTHER READING

1) Bovet, D. P. & Cesati, M. (2005). Understanding the Linux Kernel: from I/O Ports to Process Management. O'Reilly Media, Inc.
2) Burgess, M. (2004). *Principles of Network & System Admin*. John Wiley & Sons.
3) Das, S. (2000). *UNIX, Concepts and Applications*. Tata McGraw-Hill.



14.13 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: (b) **Ans. to Q. No. 2:** (d)

Ans. to Q. No. 3: (a) **Ans. to Q. No. 4:** (d)

Ans. to Q. No. 5: (c) **Ans. to Q. No. 6:** (b)

Ans. to Q. No. 7: (b) **Ans. to Q. No. 8:** (a)

Ans. to Q. No. 9: (d) **Ans. to Q. No. 10:** (d)

Ans. to Q. No. 11: (d) **Ans. to Q. No. 12:** (d)

Ans. to Q. No. 13: (c) **Ans. to Q. No. 14:** (c)



14.14 MODEL QUESTIONS

Q.1: Define network protocol?

Q.2: What is NFS? State the purpose of using NFS. Mention its advantages and disadvantages.

Q.3: State the functions of NIS. Explain the different components of NIS.

Q.4: Write the configuration of NFS server and NFS client in Linux.

Q.5: Mention the steps to configure an NIS server and NIS clients.

Q.6: Why do we use a slave server? How is it different from a master server?

Q.7: Explain the working of DNS with a suitable diagram.

Q.8: Describe the functions of FTP.

Q.9: Write the steps to configure and exchange files in Linux between a server and a client using FTP/ SFTP.

Q.10: Why do we use proxy server? State the advantages of using squid proxy server.

Q.11: Describe the following:

a) DHCP server

b) Iptables and firewalls

*** ***** ***

UNIT 15: BASIC NETWORK SECURITY ISSUES

UNIT STRUCTURE

15.1 Learning Objectives

15.2 Introduction

15.3 Network Security Issues

- 15.3.1 Firewall
- 15.3.2 TCP-IP Security

15.4 Packet Sniffing

15.5 Denial of Service

- 15.5.1 Prevention

15.6 Kernel Firewalling

- 15.6.1 Managing the Firewall Filters
- 15.6.2 Masquerading

15.7 Virtual Private Network

- 15.7.1 OpenVPN

15.8 Let Us Sum Up

15.9 Further Reading

15.10 Answers to Check Your Progress

15.11 Model Questions

15.1 LEARNING OBJECTIVES

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

- describe the basic issues of Network securities
- describe the securities of web services
- define Firewall, Eavesdropping
- describe different sets of security rules
- describe various types of security threats in the computer network
- describe security measures of TCP-IP
- explain the need of Secure Shell (ssh)
- describe the concept of packet sniffing
- explain the Denial of Service (DoS) attack and its preventive measures

- describe about Kernel Firewalling in Linux and Virtual private network (VPN).

15.2 INTRODUCTION

In the previous unit, we have learned about network protocols like network file system (NFS), network information system (NIS), domain name system (DNS) and file transfer protocol (FTP) in detail. This unit gives an introduction of different security issues in computer network. You will be able to understand the various security measures taken against the security threats. This unit gives a brief idea about the security issues related to the web services. It also discusses about the basic concept of firewall and its types. You will be able to learn about security measures of TCP-IP and ssh by going through the unit and can understand the reason of Denial of Service (DoS) attack and its preventive measures. Kernel Firewalling in Linux is also a topic of discussion in this unit in addition to the fundamental concept of Virtual private network.

15.3 NETWORK SECURITY ISSUES

One of the important services available with Linux is the Web server. The web server provides the ability to set up web sites. One of the web servers available is Apache web server. To add security to the websites the following measures need to be adopted:

Authorization: The process of determining whether a client is allowed to access your website.

Access Control: The process of allowing or preventing access to specific website objects, based on the authorization of a client

Host Based Authorization: Controlling access to the website based on the hostname or IP address.

User Based Authorization: Controlling access to the website based on the identity of a user.

Authentication: The process of determining the identity of a remote user. It involves providing a name and password to a user, which will only be known to a user.

Another key ingredient in the overall security of the website is the secure communication (that is, encryption). It is important to make the communication between the client and the web server private to provide security. This is done by secure socket layer(SSL) protocol. SSL is a protocol to allow two machines communicating over TCP-IP to encrypt the information sent between them. After a communication session is secured in this way, the two machines can exchange private or sensitive information without worrying about eavesdroppers or other interlopers stealing or using the information. Eavesdropping is the act of secretly or stealthily listening to the private conversation or communications of others without their consent. The practice is widely regarded as unethical, and in many jurisdictions it is illegal.

When communicating with a secure web server, the client must use a protocol called HTTPS (or secure HTTP). The default TCP port of HTTP protocol is 80, the default port for HTTPS is 443. When a browser tries to access a secure server at the wrong port, the browser appears to hang, and eventually times out.

Several special files are required by the server for it to operate in secure mode. The following files are used for server security:

- *A server key file:* This file contains a public and private key, which are used by the server for encryption and decryption operation.
- *A certificate file:* This file specifies that the key and website are run by a certain organization. If this certificate is designed by a trusted agency, the user can trust that the website is indeed run by indicated organization.
- *A certificate signing request:* This file contains information from the certificate, as well as information about the key. It is intended to be sent to a trusted agency (called a Certificate Authority) for signing.

Linux systems are compiled with TCP Wrappers. It is a library that provides simple access control and standardized logging for supported applications that accept connections over a network. It is a host-based networking access control list (ACL) system and is used to filter network access to Internet. TCP Wrappers present one strong advantage over firewall– it works on the application layer. It can filter requests when encryption

is used. Common services such as pop3, ftp, sshd, telnet, and r-services are supported by TCP Wrappers.

15.3.1 Firewall



Accept: allow the traffic.

Reject: block the traffic but reply with an "unreachable error".

Drop: block the traffic with no reply.

Firewall is a network security device, either hardware or software based, which monitors all incoming and outgoing traffic and based on defined set of security rules it ***accepts***, ***rejects*** or ***drop***s that specific traffic.

Firewall establishes a barrier between secured internal networks and outside un-trusted network, such as Internet.

Types of Firewalls are mentioned below:

- **Packet Filters**
- **Application Gateways**
- **Stateful Inspection Firewalls**
- **Circuit-Level Gateways**

- **Packet Filters:** It works in the network layer of the OSI Model. It applies a set of rules (based on the contents of IP and transport header fields) on each packet and based on the outcome, decides to either forward or discard the packet.

  For example, a rule could specify to block all incoming traffic from a certain IP address or disallow all traffic that uses UDP protocol. If there is no match with any predefined rules, it will take default action. The default action can be to 'discard all packets' or to 'accept all packets'.

  Security threats to packet filters:

  IP address Spoofing: In this kind of attack, an intruder from the outside tries to send a packet towards the internal network with the source IP address setting equal to one of the IP addresses of internal users.

  Source Routing Attacks: In this kind of attack, the attacker specifies the route to be taken by the packet with a hope to fool the firewall.

Tiny Fragment Attacks: Many times, the size of the IP packet is greater than the maximum size allowed by the underlying network such as Ethernet, Token Ring etc. In such cases, the packet needs to be fragmented, so that it can be carried further. The attacker uses this characteristic of TCP/IP protocol. In this kind of attack, the attacker intentionally creates fragments of the original packet and sends it to fool the firewall.

Preventive measures for these type of threats:

- Firewall can protect from **IP address spoofing,** if it discards all the packets that arrive at the incoming side of the firewall, with source IP equal to one of the internal IPs.
- Firewall can protect **Source Routing** attack if it discards all the packets that use the option of source routing.
- Firewall can defeat **Tiny Fragment Attacks** if it discards all the packets which use the TCP protocol and is fragmented.

- **Application Gateways:** It is also known as **Proxy server**. It works as follows:

  Step 1: User contacts the application gateway using a TCP/IP application such as HTTP.

  Step 2: The application gateway asks about the remote host with which the user wants to establish a connection. It also asks for the user id and password that is required to access the services of the application gateway.

  Step 3: After verifying the authenticity of the user, the application gateway accesses the remote host on behalf of the user to deliver the packets.

- **Stateful Inspection Firewalls:** It is also known as 'Dynamic Packet Filters'. It keeps track of the state of active connections and uses this information to decide which packets to be allowed through it, i.e., it adapts itself to the current exchange of information, unlike the normal packet filters/stateless packet filters, which have hardcoded routing rules.

- **Circuit-Level Gateways:** It acts as a virtual connection between the remote host and the internal users by creating a new connection between itself and the remote host. It also changes the source IP address in the packet and puts its own address at the place of source IP address of the packet from the end users. This way, the IP addresses of the internal users are hidden and secured from the outside world.

All data flowing to and from the Internet and the private network is filtered by the firewall. Inside the private network, less care needs to be taken with turning off services. It is a way of concentrating effort on making one machine secure and protecting many others in the process. To make a set up, first it is required to configure the machine for two Ethernet cards. Then, it needs to use the IP-chains/IP-tables software to set up filters which connect the two Ethernet cards data links.

The main drawback with making the systems more secure is that they become less accessible. The idea behind increasing the system's security is to stop the use of the computers by crooks, thieves and malcontents. Linux has a wide range of security tools and by carefully combining various techniques and programs.

15.3.2 TCP-IP Security

The TCP-IP suite of protocols is remarkably robust, but it has no real provisions for security (i.e., authentication, verification, encryption and so on). Spoofing packets, intercepting packets, reading data payloads frequently occur in Internet. The most common attacks are denial of service attacks since they are the easiest to execute and the hardest to defeat, followed by packet sniffing, port scanning, and other related activities.

While data is in transit between programs over the Internet, it is also in danger. Anyone with the knowledge can look at the data. Using a method known as 'spoofing', they can even inject fake data into a legitimate stream. These problems come about because of

the way by which Internet protocols interact. To overcome these difficulties, ssh (Secure Shell) was devised.

Ssh is a stable, well-developed system with open source that provides encryption and authentication on connections. Encryption process uses codes to protect the packets of data while in transit. Authentication is a process for verifying if a packet of data or a connection is valid. By using Linux as a server, ssh level security can be provided for all the network use.

Ethereal is a packet sniffer. It is used to capture various types of packets over a given period of time. It also shows all manner of information about the packets. It is useful for watching packets coming into and going out of your machine. Generally it will detect traffic on the network segment.

Another logging/intrusion detection type tool is called *Tripwire*. It takes a snapshot of the important system files and records their signature in a database. Various signature levels are available from mild to wild. In a customized way rules can be set in a policy file to tell Tripwire, what to check. After the database is initialized and signed, Tripwire can be executed whenever it is needed to check the integrity of the system. The report will point out when the files are changed and the severity of the security risk. The Tripwire report is easy to read and can be customized according to the file tracking needs.

A popular program for detecting access attempts (via the network) and port scans is *Snort*. The program produces files that log these types of activities and even gives some idea of where to find out more information.



CHECK YOUR PROGRESS

Q.1: What is Host Based Authorization?

.....

.....

Q.2: The security rule **Reject means** block the traffic but reply with ..... error.

Q.3: In which layer packet filters works?

.....

15.4 PACKET SNIFFING

When any data has to be transmitted over the computer network, it is broken down into smaller units at the sender's node called data packets and reassembled at receiver's node in original format. It is the smallest unit of communication over a computer network. It is also called a block, a segment, a datagram or a cell. The act of capturing data packet across the computer network is called packet sniffing. It is similar to as wire tapping to a telephone network. It is mostly used by crackers and hackers to collect information illegally about network. It is also used by ISPs, advertisers and governments.

ISPs use packet sniffing to track some of the activities such as:

- Who is receiver of your email.
- What is the content of that email.
- What you download.
- Sites you visit.
- What you looked on that website.
- Downloads from a site.
- Streaming events like video, audio, etc.

Advertising agencies or internet advertising agencies are paid according to:

- number of ads shown by them.
- number of clicks on their ads also called PPC (pay per click).

To achieve this target, these agencies use packet sniffing to inject advertisements into the flowing packets. Most of the time, these ads contain malware.

Government agencies use packet sniffing to:

- Ensure security of data over the network.
- Track an organization's unencrypted data.

Packet Sniffer: Packet sniffing is done by using tools called packet sniffer. It can be either filtered or unfiltered. Filtered is used when only specific data packets have to be captured and unfiltered is used when all the packets have to be captured. WireShark, SmartSniff are examples of packet sniffing tools.

How to prevent packet sniffing:

- Encrypting data you send or receive.
- Using trusted Wi-Fi networks.
- Scanning your network for dangers or issues.

15.5 DENIAL OF SERVICE (DOS)

Denial of Service (DoS) is a cyber-attack on an individual computer or website with an intent to deny services to intended users. Their purpose is to disrupt an organization's network operations by denying access to its users. Denial of service is typically accomplished by flooding the targeted machine or resource with surplus requests in an attempt to overload systems and prevent some or all legitimate requests from being fulfilled.

For example, if a bank website can handle 10 people in a second by clicking the Login button, an attacker only has to send 10 fake requests per second to make it so such that no legitimate users can login.

DoS attacks exploit various weaknesses in computer network technologies. They may target servers, network routers, or network communication links. They can cause computers and routers to crash and link to bog down.

The most famous DoS technique is Ping of Death. The Ping of Death attack works by generating and sending special network messages (specifically, ICMP packets of non-standard sizes) that cause problems for systems that receive them. In the early days of the Web, this attack could cause unprotected Internet servers to crash quickly.

The following command is used for performing flooding of request on an IP:

```
ping ip_address –t -65500
```

Here,

- "ping" sends the data packets to the victim.
- "ip_address" is the IP address of the victim.
- "-t" means the data packets should be sent until the program is stopped.
- "-(65500)" specifies the data load to be sent to the victim.

Other basic types of DoS attacks involve:

- Flooding a network with useless activity so that genuine traffic cannot get through. The TCP/IP SYN and smurf attacks are two common examples.
- Remotely overloading a system's CPU so that valid requests cannot be processed.
- Changing permissions or breaking authorization logic to prevent users from logging into a system. One common example involves triggering a rapid series of false login attempts that lock out accounts from being able to log in.
- Deleting or interfering with specific critical applications or services to prevent their normal operation (even if the system and network overall are functional).

Another variant of the DoS is the Smurf_attack. This involves emails with automatic responses. If someone emails hundreds of email messages with a fake return email address to hundreds of people in an organization with an autoresponder on in their email, the initial sent messages can become thousands sent to the fake email address. If that fake email address actually belongs to someone, this can overwhelm that person's account.

DoS attacks can cause the following problems:

- Ineffective services
- Inaccessible services
- Interruption of network traffic
- Connection interference.

15.5.1 Prevention

Denial of Service (DoS) attacks is becoming more frequent and therefore it needs to review the basics of DoS.

- Cloud Mitigation Provider– Cloud mitigation providers are experts at providing DoS mitigation from the cloud. This means they have built out massive amounts of network bandwidth and DoS mitigation capacity at multiple sites around the Internet that can take in any type of network traffic, whether you use multiple ISPs, your own data center or any number of cloud providers. They can scrub the traffic for you and only send "clean" traffic to your data center.
- Firewall– This is the simplest and least effective method. Generally someone writes some Python scripts that try to filter out the bad traffic or an enterprise will try and use its existing firewalls to block the traffic
- Internet Service Provider (ISP)– Some enterprises use their ISP to provide DoS mitigation. These ISPs have more bandwidth than an enterprise would, which can help with the large volumetric attacks.



CHECK YOUR PROGRESS

Q.4: How to prevent Denial of Service (DoS) attacks?

.....

.....

Q.5: What is packet sniffing?

.....

.....

Q.6: How to prevent packet sniffing?

.....

.....

.....

.....

15.6 KERNEL FIREWALLING

The Linux kernel provides native support for IP packet filtering at several stages: when a packet is received, when packet is sent, and when a packet is being forwarded. Each of the three filters consists of a default policy and a list of filter rules. Every filter rule defines some packet characteristics, like IP addresses, an optional network device, and several other options. Furthermore, each rule has a policy associated with it, defining what to do when a packet matches with the rule,

The algorithm used in the filters can be described as follows:

Step 1: Step through the list of filter rules associated with the filter and check whether the packet matches with the rule or not.

Step 2: The first matching filter rule (if any) determines all further actions:

- The rule's policy will be applied to the packet.
- Each rule contains packet and byte counters, which will be incremented when a packet matches.
- Optionally, some information about the packet is written to the Linux kernel log.
- Finally, a rule may contain parameters defining how to change the *TOS-field* in the IP header, dealing with the packet's priority.

Step 3: If none of the filter rules match with the packet, use the default policy associated with the filter.

Filter rules in Linux contain the following items:

- IP source and destination addresses, both with their own 32-bit mask. A mask containing only 0's matches with every address.
- The protocol, being either TCP, UDP, ICMP, or "any".
- Source and destination port numbers (services), used in combination with TCP or UDP packets. Up to ten source and destination ports may be specified in one rule, including ranges of ports (like 1024-65535, representing all unprivileged ports).
- Message types, used with ICMP packets.

- Bits to match with the TCP flags ACK and SYN, used to refuse the establishment of new TCP connections in a certain direction.
- The name or IP address of a network device. Rules containing a device specification will only match with packets coming in (or going out) via that particular device.
- A specification for changing the *TOS-field* in the IP header, being used when a packet is accepted by that rule.
- A flag indicating if some basic packet information should be written to the Linux kernel log, in case a packet matches with that rule.

15.6.1 Managing the Firewall Filters

The interface for managing the kernel-level filter rules at user-level mainly consists of two parts:

- Firewall rules can be created or deleted via the setsockopt(2) system call. Via the same mechanism the default policy of a filter can be changed.
- Filters can be inspected by reading the following pseudo-files in the /proc file system:

  /proc/net/ip_input

  /proc/net/ip_output

  /proc/net/ip_forward

Each of these files lists the default policy, followed by the details of all rules (if any) belonging to that filter, in a compact format. The *ipfwadm* command provides a command-level interface for managing the Linux firewall facilities: it can be used to change or inspect all aspects of the kernel filters. Let's start with a simple example:

```
ipfwadm -I -a deny -S 192.168.22.0/24 -D 0.0.0.0/0
```

This command basically means "refuse all incoming packets originally coming from network 192.168.22.0''. It appends (-a) a new rule to the list of filter rules belonging to the *input* firewall (-I). The

output and *forward* filters can be changed by using the -O and -F options, respectively. Valid keywords are accept, deny, and reject (refuse the packet, but return an ICMP message). The source (-S) and destination (-D) addresses both include a mask: the suffixes /24 and /0 are equivalent to /255.255.255.0 and /0.0.0.0, respectively. Every IP address will match the specified destination, because the mask only contains 0's.
Another example:

```
ipfwadm -I -a accept -k -P tcp -S 0.0.0.0/0 telnet
\ -D 192.168.37.1 1024:65535

ipfwadm -O -a accept -P tcp -S 192.168.37.1 1024:65535
\ -D 0.0.0.0/0 telnet
```

The command creates two rules (one for the input firewall, one for the output firewall) that accepts all packets belonging to an outgoing telnet connection (here it is assumed that the local IP address is 192.168.37.1). The protocol is specified via the -P option and after the IP address, a service name (telnet, specifying port 23) and a port range are specified. The -k option makes the input rule only match with packets having the TCP ACK flag set. This prevents someone from trying to initiate a connection from the outside (using source port 23) to some unprivileged port on the system.

15.6.2 Masquerading (Transparent Proxying)

The Linux kernel provides an additional mechanism to use in firewall solutions: masquerading of IP packets. This means that some or all packets being forwarded by a Linux system can be changed as if they were sent from the local system. So, the source IP address is replaced by the local IP address and the source port is replaced by a locally generated port (e.g., 60005). Because an administration is kept of masqueraded sessions, incoming packets for that port will automatically be "demasqueraded" and forwarded to the system that originally initiated the session.

The next table summarizes the masquerading function; given a telnet session from an internal host (192.168.37.15) to an external host (10.42.17.8), passing a Linux system doing masquerading (192.168.37.1):

	Source		Destination	
	IP address	**Port**	***IP address***	**Port**
original packet	192.168.37.15	1027	10.42.17.8	23
masqueraded	192.168.37.1	60005	10.42.17.8	23
reply packet	10.42.17.8	23	192.168.37.1	60005
demasqueraded	10.42.17.8	23	192.168.37.15	1027

One of the problem areas is found in the widely used *ftp* protocol, because this protocol uses a second session (normally initiated by the remote site) for transferring the actual data. A similar problem arises with the IRC protocol. The Linux IP masquerading implementation deals with such protocol-specific features in separately loadable modules. Another problem is that a transparent proxy should operate on transport level connections, whereas masquerading is implemented in the network layer. The current implementation tries to address this with a limited session administration, but there are still some weaknesses to work on.

15.7 VIRTUAL PRIVATE NETWORK (VPN)

VPN networks are often operated as client-server applications. VPN server is directly running on a firewall, where it creates virtual network interface and additional virtual network subnet. VPN server is waiting for connections on the external network interface of the firewall where it performs authentication of a VPN client application. After successful VPN client authentication a VPN client is given an IP address from a virtual subnet. Consequently, an encrypted tunnel is created between VPN client and VPN server, which is used for safe transfer of packets between two distant networks via the Internet. Services, which a VPN client can connect to, can furthermore be defined by firewall rules. This way firewall ensures that VPN client can connect only to services it is allowed to connect.

15.7.1 OpenVPN

OpenVPN is a piece of software dedicated to creating virtual private networks. Its setup involves creating virtual network interfaces on the VPN server and on the client(s); both **tun** (for IP-level tunnels) and **tap** (for Ethernet-level tunnels) interfaces are supported. In practice, **tun** interfaces will most often be used except when the VPN clients are meant to be integrated into the server's local network by way of an Ethernet bridge.

OpenVPN relies on OpenSSL for all the SSL/TLS cryptography and associated features (confidentiality, authentication, integrity, non-repudiation). It can be configured either with a shared private key or using X.509 certificates based on a public key infrastructure. The latter configuration is strongly preferred since it allows greater flexibility when faced with a growing number of roaming users accessing the VPN.



CHECK YOUR PROGRESS

Q.7: What is VPN?

.....

.....

.....

Q.8: What is Masquerading?

.....

.....

.....

.....

Q.9: What is ipfwadm command?

.....

.....

.....



15.8 LET US SUM UP

- To add security to the websites following measures needs to be adopted: Authorization, Access Control, Host Based Authorization, User Based Authorization, and Authentication.
- Packet filters works in the network layer of the OSI Model. It applies a set of rules (based on the contents of IP and transport header fields) on each packet and based on the outcome, decides to either forward or discard the packet.
- Ssh is a stable, well-developed system with open source that provides encryption and authentication on connections.
- The act of capturing data packet across the computer network is called packet sniffing.
- Denial of Service (DoS) is a cyber-attack on an individual Computer or Website with an intent to deny services to intended users. Their purpose is to disrupt an organization's network operations by denying access to its users.
- The Linux kernel provides an additional mechanism to use in firewall solutions: masquerading of IP packets. This means that some or all packets being forwarded by a Linux system can be changed as if they were sent from the local system.
- VPN networks are often operated as client-server applications. VPN server is directly running on a firewall, where it creates virtual network interface and additional virtual network subnet.



15.9 FURTHER READING

1) Sobell, M. G. & Sobell, M. G. (1995). *A Practical Guide to the Unix System*. Benjamin/Cummings.
2) Tackett, J., Burnett, S. & Koch, J. (1999). *Special Edition Using Linux, 5E*. Macmillan Publishing Co., Inc.



15.10 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q. No. 1: It is a process of Controlling access to the website based on the hostname or IP address.

Ans. to Q. No. 2: The security rule **Reject means** block the traffic but reply with an unreachable error.

Ans. to Q. No. 3: It works in the network layer of the OSI Model.

Ans. to Q. No. 4: Cloud Mitigation Provider, Firewall and ISP provide DoS mitigation.

Ans. to Q. No. 5: The act of capturing data packet across the computer network is called packet sniffing.

Ans. to Q. No. 6: By adopting following steps packet sniffing is prevented:

i) Encrypting data you send or receive.

ii) Using trusted Wi-Fi networks.

iii) Scanning your network for dangers or issues.

Ans. to Q. No. 7: VPN networks are often operated as client-server applications. VPN server is directly running on a firewall, where it creates virtual network interface and additional virtual network subnet.

Ans. to Q. No. 8: The Linux kernel provides an additional mechanism to use in firewall solutions: masquerading of IP packets. This means that some or all packets being forwarded by a Linux system can be changed as if they were sent from the local system.

Ans. to Q. No. 9: The *ipfwadm* command provides a command-level interface for managing the Linux firewall facilities: it can be used to change or inspect all aspects of the kernel filters.



15.11 MODEL QUESTIONS

Q.1: Discuss the different types of security threats faced in the computer network?

Q.2: What is packet sniffing? How does it work?

Q.3: What is VPN? Discuss the basic structure of VPN.

Q.4: What is the basic purpose of Firewall?

Q.5: Discuss the different types of preventive measures to prevent DoS attack.

Q.6: What is the use of masquerading of IP packets?

Q.7: How do Stateful Inspection Firewalls work?

Q.8: What is kernel Firewalling?

Q.9: Discuss the functionality of Application Gateways.

Q.10: How can Web servers be protected from security threats?

*** ***** ***



REFERENCES

1) Bandel, D. & Napier, R. (2000). *Special Edition Using Linux*. Que Corp.
2) Bovet, D. P. & Cesati, M. (2005). *Understanding the Linux Kernel: from I/O Ports to Process Management*. O'Reilly Media, Inc.
3) Burgess, M. (2004). *Principles of Network & System Admin*. John Wiley & Sons.
4) Das, S. (2012). *Your UNIX/Linux: The Ultimate Guide*. Science/ Engineering/Math. McGraw-Hill.
5) Das, S. (2000). UNIX, *Concepts and Applications*. Tata McGraw-Hill.
6) Hahn, H. (2008). *Harley Hahn's Guide to Unix and Linux*. California: McGraw-Hill Higher Education.
7) Kanetkar, Y. P. (2003). *UNIX Shell Programming*. BPB Publications.
8) Negus, C. (2012). *Linux Bible* (Vol. 772). John Wiley & Sons.
9) Negus, C. (2003). *Red Hat Linux 9 Bible*. John Wiley & Sons, Inc.
10) Petersen, R. (2007). *Linux: the Complete Reference*. McGraw-Hill, Inc.
11) Siever, E., Weber, A., Figgins, S., Love, R. & Robbins, A. (2005). *Linux in a Nutshell*. O'Reilly Media, Inc.
12) Sobell, M. G. & Sobell, M. G. (1995). *A Practical Guide to the Unix System*. Benjamin/Cummings.
13) Sobell, M. G. (2009). *A Practical Guide to Linux Commands, Editors, and Shell Programming*. Pearson Education.
14) Tackett, J., Burnett, S. & Koch, J. (1999). *Special Edition Using Linux, 5E*. Macmillan Publishing Co., Inc.

Name of the Paper: .....
Course Code: .....

Centre for Internal Quality Assurance (CIQA)
Krishna Kanta Handiqui State Open University
City Office: Housefed Complex, Guwahati-781006

Learner's Feedback on Course

Dear Learner,

Regarding the course as mentioned above, we would like to know your opinions and comments so as to improve the quality of self learning materials in future. Please respond to the following statements by ticking the number you feel most reflect your opinion. After completion of the additional comments, please detach the page and send/mail the same to us at the address given below.

The Director, Centre for Internal Quality Assurance, KKHandiqui State Open University
Housefed Complex, Dispur, Guwahati-781006
(E-mail id: ciqa@kkhsou.in)

1) Approximately how many hours did you spend for studying the units in the course? ☐☐☐

2) Please give your opinions (by ✓ mark) to the following items based on your reading of the block:

Sl. No.	Statements	Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
I)	The SLMs of the course fulfil the learning objectives set out in the beginning of each unit					
II)	The units of the SLM could satisfy my academic needs and expectations					
III)	The Unit writers have excellent knowledge about the course contents					
IV)	Language and contents of the units were presented at a level which I could readily understand					
V)	Ample opportunity for participation in the activities provided in the units					
VI)	Used enough Illustrations (Diagrams, tables etc.) for conceptual clarity					
VII)	Quality of content is engaging, relevant, and up-to-date					
VIII)	The self check questions are very helpful					
IX)	The Possible/model questions and the answers to check my progress have benefited me a lot					

Additional Comments: *(Please feel free to provide your open comments)*

1) Which aspects of the SLM, according to you, worked well?

.....

.....

2) What sort of changes/improvements do you feel KKHSOU could implement to improve the overall quality of the SLM?

.....

.....

Thank you for taking the time to complete this form.