

# Working with Red Hat Enterprise Linux 5.0

Are you registered with  
**Onlinevarsity.com**?

Yes



No



Did you download this book  
from **Onlinevarsity.com**?

Yes



No



## Scores

For each **YES** you score **50**

For each **NO** you score **0**

If you score less than 100 this book is illegal.

Register on **[www.onlinevarsity.com](http://www.onlinevarsity.com)**

# Working with Red Hat Enterprise Linux 5.0 Learner's Guide

© 2013 Aptech Limited

All rights reserved.

No part of this book may be reproduced or copied in any form or by any means – graphic, electronic or mechanical, including photocopying, recording, taping, or storing in information retrieval system or sent or transferred without the prior written permission of copyright owner Aptech Limited.

All trademarks acknowledged.

**APTECH LIMITED**

Contact E-mail: [ov-support@onlinevarsity.com](mailto:ov-support@onlinevarsity.com)

First Edition - 2013



## Dear Learner,

We congratulate you on your decision to pursue an Aptech course.

Aptech Ltd. designs its courses using a sound instructional design model – from conceptualization to execution, incorporating the following key aspects:

- Scanning the user system and needs assessment

Needs assessment is carried out to find the educational and training needs of the learner

Technology trends are regularly scanned and tracked by core teams at Aptech Ltd. TAG\* analyzes these on a monthly basis to understand the emerging technology training needs for the Industry.

An annual Industry Recruitment Profile Survey<sup>#</sup> is conducted during August - October to understand the technologies that Industries would be adapting in the next 2 to 3 years. An analysis of these trends & recruitment needs is then carried out to understand the skill requirements for different roles & career opportunities.

The skill requirements are then mapped with the learner profile (user system) to derive the Learning objectives for the different roles.

- Needs analysis and design of curriculum

The Learning objectives are then analyzed and translated into learning tasks. Each learning task or activity is analyzed in terms of knowledge, skills and attitudes that are required to perform that task. Teachers and domain experts do this jointly. These are then grouped in clusters to form the subjects to be covered by the curriculum.

In addition, the society, the teachers, and the industry expect certain knowledge and skills that are related to abilities such as *learning-to-learn, thinking, adaptability, problem solving, positive attitude etc.* These competencies would cover both cognitive and affective domains.

**A precedence diagram for the subjects is drawn where the prerequisites for each subject are graphically illustrated. The number of levels in this diagram is determined by the duration of the course in terms of number of semesters etc. Using the precedence diagram and the time duration for each subject, the curriculum is organized.**

- Design & development of instructional materials

The content outlines are developed by including additional topics that are required for the completion of the domain and for the logical development of the competencies identified. Evaluation strategy and scheme is developed for the subject. The topics are arranged/organized in a meaningful sequence.

The detailed instructional material – Training aids, Learner material, reference material, project guidelines, etc.- are then developed. Rigorous quality checks are conducted at every stage.

➤ Strategies for delivery of instruction

Careful consideration is given for the integral development of abilities like thinking, problem solving, learning-to-learn etc. by selecting appropriate instructional strategies (training methodology), instructional activities and instructional materials.

The area of IT is fast changing and nebulous. Hence considerable flexibility is provided in the instructional process by specially including creative activities with group interaction between the students and the trainer. The positive aspects of web based learning –acquiring information, organizing information and acting on the basis of insufficient information are some of the aspects, which are incorporated, in the instructional process.

➤ Assessment of learning

The learning is assessed through different modes – tests, assignments & projects. The assessment system is designed to evaluate the level of knowledge & skills as defined by the learning objectives.

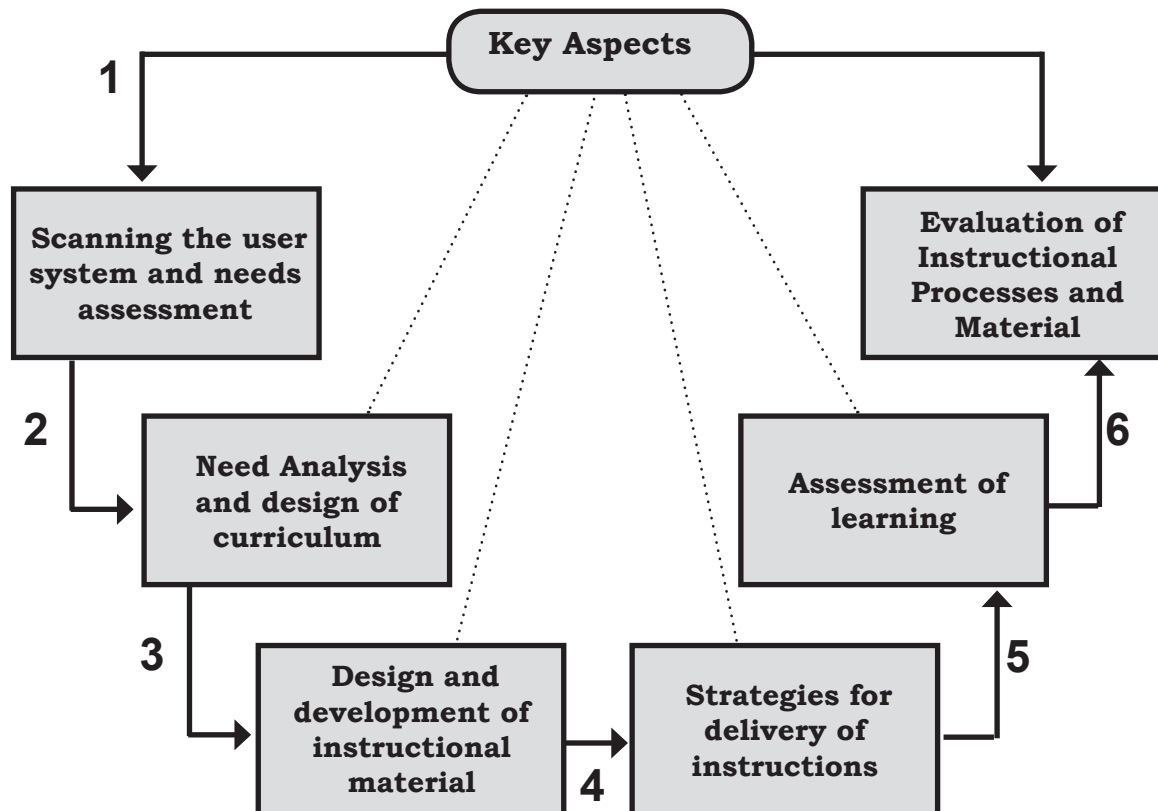
➤ Evaluation of instructional process and instructional materials

The instructional process is backed by an elaborate monitoring system to evaluate - on-time delivery, understanding of a subject module, ability of the instructor to impart learning. As an integral part of this process, we request you to kindly send us your feedback in the reply pre-paid form appended at the end of each module.

\*TAG – Technology & Academics Group comprises of members from Aptech Ltd., professors from reputed Academic Institutions, Senior Managers from Industry, Technical gurus from Software Majors & representatives from regulatory organizations/forums.

Technology heads of Aptech Ltd. meet on a monthly basis to share and evaluate the technology trends. The group interfaces with the representatives of the TAG thrice a year to review and validate the technology and academic directions and endeavors of Aptech Ltd.

## Aptech New Products Design Model





**“A little learning is a dangerous thing,  
but a lot of ignorance is just as bad”**

## Preface

---

Linux is open source operating system that makes it easily accessible. Red Hat Enterprise Linux is a fast, stable, and easy to operate system. In addition, it also provides high security to the users as the operating system is less prone to viruses. The advantage of Red Hat Enterprise Linux is that it also provides us with a graphical interface., in addition to the command prompt.

In this book, we will deal with the X Window System and learn to operate the GNOME environment. We shall also learn about the file systems available in Red Hat Enterprise Linux 5.0. This book aims at making the students comfortable with the command line by providing various command line commands. The students will also learn to work with directories at the command prompt that includes, creating, deleting, compressing and archieving directories. The course also enables the students to manage users connection to the network.

This book is the result of a concentrated effort of the Design Team, which is continuously striving to bring you the best and the latest in Information Technology. The process of design has been a part of the ISO 9001 certification for Aptech-IT Division, Education Support Services. As part of Aptech's quality drive, this team does intensive research and curriculum enrichment to keep it in line with industry trends.

We will be glad to receive your suggestions.

Design Team



The background is a grayscale, high-contrast image of a computer keyboard and a circuit board. The keyboard keys are visible in the upper half, with labels like 'CTRL', 'SHIFT', and 'BACK' partially legible. The lower half shows a detailed view of a circuit board with various components, including a large circular component and numerous small electronic components. The overall aesthetic is technical and modern.

**“ Nothing is a waste of time if you  
use the experience wisely ”**



---

## Table of Contents

---

### Sessions

1.	Working with the X Window System	1
2.	Working with the X Window System (Lab)	21
3.	Working with the File System and Command Line	45
4.	Working with the File System (Lab)	67
5.	The bash Shell and bash Shell Scripting	77
6.	Shell Scripting (Lab)	95
7.	Networking	105
8.	Networking (Lab)	123



**“ Learning is not compulsory  
but neither is survival ”**

# Session 1

## Working with the X Window System

### Session Objectives:

*At the end of this session, you will be able to -*

- Describe the features of Red Hat Enterprise Linux 5.0
- Explain how to log on to the Linux system
- Explain the root user and shadow passwords
- Explain the Desktop environment and Window's Managers
- Explain the concept of users and groups
- Explain the use of some basic and advanced Linux commands

### 1.1 Introduction to Linux and the X Window System

---

Linux is a powerful, server based operating system. The main features of Linux are: high speed, efficiency, and flexibility. The X Window System (X) is a dominant graphical user interface (GUI) for Linux. Different versions of X Window System are available for other operating systems. The X Window System provides the foundation for Window Managers such as the K Desktop Environment (KDE) and GNU's Not Unix (GNU) Network Object Model Environment (GNOME) in Linux.

#### 1.1.1 Introduction to Red Hat Enterprise Linux 5.0

Red Hat Enterprise Linux (RHEL) is a Linux operating system developed for the business market. RHEL was formerly known as Red Hat Linux Advanced Server. The features of Red Hat Enterprise Linux 5.0 are as follows:

- RHEL 5 is available in six versions. They are as follows:
  - ✧ RHEL (formerly called ES, which was sometimes said to stand for Economy or Entry-level Server)
  - ✧ RHEL Advanced Platform (formerly called AS, which was sometimes said to stand for Advanced or Application Server)
  - ✧ RHEL Desktop
  - ✧ RHEL Desktop with Multi-OS option
  - ✧ RHEL Desktop with Workstation and Multi-OS option
  - ✧ RHEL Desktop with Workstation option (former WS)
- The RHEL 5 distribution package includes:
  - ✧ Linux kernel 2.6.18
  - ✧ Apache 2.2.3
  - ✧ MySQL 5.0.22
  - ✧ PHP 5.1.6
  - ✧ PostgreSQL 8.1.4

## Working with Red Hat Enterprise Linux 5.0

- Red Hat Enterprise Linux 5 server: There are two versions of Red Hat Enterprise Linux available for running on the server. They are as follows:
  - ✧ A base Red Hat Enterprise Linux server, designed for small deployments.
  - ✧ Red Hat Enterprise Linux Advanced Platform designed to provide the most cost-effective, flexible, and scalable environment.
- Red Hat Enterprise Linux 5 Desktop: This is an alternative for client systems like desktop and laptop computers. RHEL Desktop environment focuses on the areas of security and management.

### 1.1.2 Logging into and Logging out of the Linux System

To access a Linux based system, the user must be authenticated. Authentication of the user is done through a login process. A login process involves entering a valid username and password. The user can log in to the system console in one of the following two ways:

- A text based login
- A GUI based login

In either case, the user will be prompted to enter the login name and password. If either of them is entered incorrectly, the user cannot logon to the system. If both the login name and password are entered correctly, the user will be logged in.

If the system is text-based, a command prompt, normally ending with a dollar sign (\$) is displayed.

```
localhost login:student
Password:
Last login:Mon Sep 15 14:30:46
```

As seen above, the password that is typed is invisible to the user.

If the system boots directly into the X Window System, the display seen by the user depends on the display manager being used. The default display manager for Red Hat Linux is `gdm`, the GNOME Display Manager.

Virtual consoles is a combination of the keyboard and the display for a user interface. It allows the user to have multiple login even when the user is not using the X Window System. Virtual consoles provide a full screen, non graphical access to the Linux system. A Linux system can run six virtual consoles and one graphical console at a time. The user can switch between the virtual consoles by pressing the key combination **Ctrl+Alt+F[1-6]**. The graphical console can be accessed by pressing the key combination **Ctrl+Alt+F7**.

The login session can be ended by typing the `logout` command at the shell prompt. This command logs the user out of the system, and a new `login:` prompt appears on the terminal.

```
[student@localhost ~]$ logout
localhost login:
```

### 1.1.3 Desktop Environment and Window Manager

The X Window System provides a GUI for Linux. It is portable and is a network transparent client/server interface between the hardware and the desktop environment. The Red Hat Enterprise Linux 5.0 uses the X11R7.1 release as the base X Window System.

While the Window Manager customizes the desktop according to the needs of the user, the desktop environment is a user interface that runs on the Window Manager. The X Window System provides a place to position the windows but does not control the access to it. So, to control the windows, an additional software, called Window Manager, is needed.

#### ➤ Desktop Environment

A desktop environment is used to create a common graphical user environment by combining various X clients. The

two types of desktop environments provided by Red Hat Enterprise Linux 5.0 are as follows:

- ✧ **GNOME:** It is the default desktop environment and can be run on multiple operating systems. The screen of the GNOME interface consists of the GNOME Panel and the desktop area. The GNOME Panel holds all GNOME applications.
- ✧ **KDE:** K Desktop Environment (KDE) is a powerful graphical desktop environment for Linux. KDE provides a complete desktop environment including a Window Manager and a large number of X utilities. It uses K Windows Manager (KWM) as the default Window Manager.

KDE and GNOME are fully operational desktop environments supporting drag-and-drop operations. Both, KDE and GNOME, rely on the underlying X Window System. It is possible to install both GNOME and KDE and switch applications from one desktop environment to another.

Linux users can use both, the command line interface and a graphical icon that is present on the desktop to switch between GNOME and KDE environments.

### ➤ Window Manager

The Window Manager is used to control the working of the desktop. It helps in controlling the windows, that is, moving, hiding, resizing, iconifying, and closing them.

The different types of Window Managers provided by Red Hat Enterprise Linux 5.0 are as follows:

- ✧ **kwin** : The kwin is the default Window Manager for KDE that supports custom themes.
- ✧ **metacity** : The Metacity is the default Window Manager for GNOME that supports custom themes; requires installation of the metacity package.
- ✧ **mwm** : The Motif Window Manager is a stand-alone Window Manager; it requires installation of the `openmotif` package.
- ✧ **twm** : The Tab Window Manager can be used as a stand-alone window manager as well as with the desktop environment. It is available with X11R7.1 release.

## 1.2 Users and Groups

---

To access the Linux system, the user has to logon to the system. The accounts that exist on the system are termed as users. The users from the organization often need to share common files amongst them. For this, the concept of groups has been introduced in Linux. This will enable the users of the groups to read, write and execute common files.

Every user and group logging into the system needs to be uniquely identified by the system. This is done using unique numerical identification numbers called **userid** and **groupid**. The user who creates the file is called the owner and group owner of the file. Each file is assigned different read, write, and execute permissions for individual users and groups. The access permissions for the file are assigned by the root user or the file owner.

### 1.2.1 The root user

The root user is a special administrative account, that has unrestricted access to all the files, devices, and programs in the system. The root user, also known as the **superuser**, guards the system against accidental damages.

### 1.2.2 Shadow Passwords

Systems are vulnerable to brute force attacks, that is, an attacks that require trying all (or a large fraction of all) possible values until the right value is found. The passwords must be protected from these brute force attacks. Shadow passwords help in improving the security of the system authentication files. In this, the passwords are hashed and hidden from the unauthorized users. This is done by placing the passwords in the `/etc/shadow` file which can be read only by the root user.

Shadow passwords help to track the duration for which an account has been inactive.

### 1.2.3 Managing Users

When a user logs into the system, Linux automatically places the user in a directory called the **home** directory. The home directory is created by the system when a user account is created. If the user with login name **wilson** logs in, a directory is created for the user having the pathname **/usr/wilson** or **/home/wilson**. The home directory is decided by the system administrator at the time of creating a user account, and the details are stored in the file **/etc/passwd**. The **/etc/passwd** file is an ASCII file that contains an entry for each user. The difference between the **/etc/passwd** and **/etc/shadow** files is that the shadow file does not have general read permission. Only the root user has the necessary permission to read and write to the shadow files.

When a new user is added to the system, a private user group, having the same name as the user name, is created with the same name as the user's name. A graphical program, called **User Manager**, is included in the Red Hat Enterprise Linux for managing the users and groups. This program is started by selecting the **Users and Groups** option from the **Administration** menu. This invokes the **User Manager** window. All the existing users are listed in the **Users** tab of this window as shown in figure 1.1.

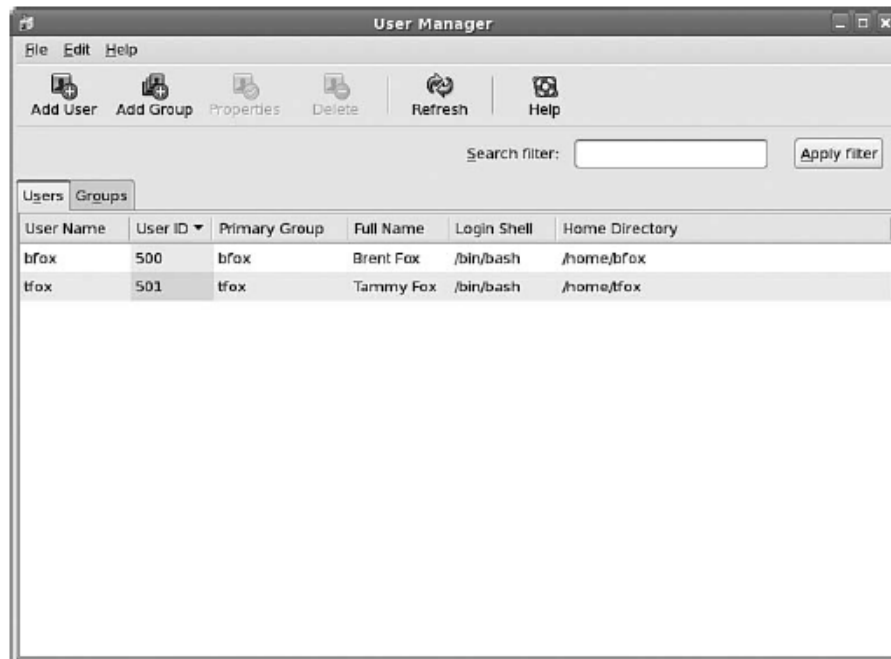


Figure 1.1: List of Existing Users

A new user can be added to the system by clicking the **Add User** button in the **User Manager** window. The **Create New User** window is displayed as shown in figure 1.2. The details of the new user can be entered into this dialog box. After the details have been entered and the **OK** button clicked, the new user is created and listed in the **User Manager** window. The default login shell for the new user is **bash**. A directory named **/home/<username>** is automatically created as a default home directory for the new user. A user can be deleted by clicking the **Delete** button.



The 'Create New User' dialog box contains the following fields and options:

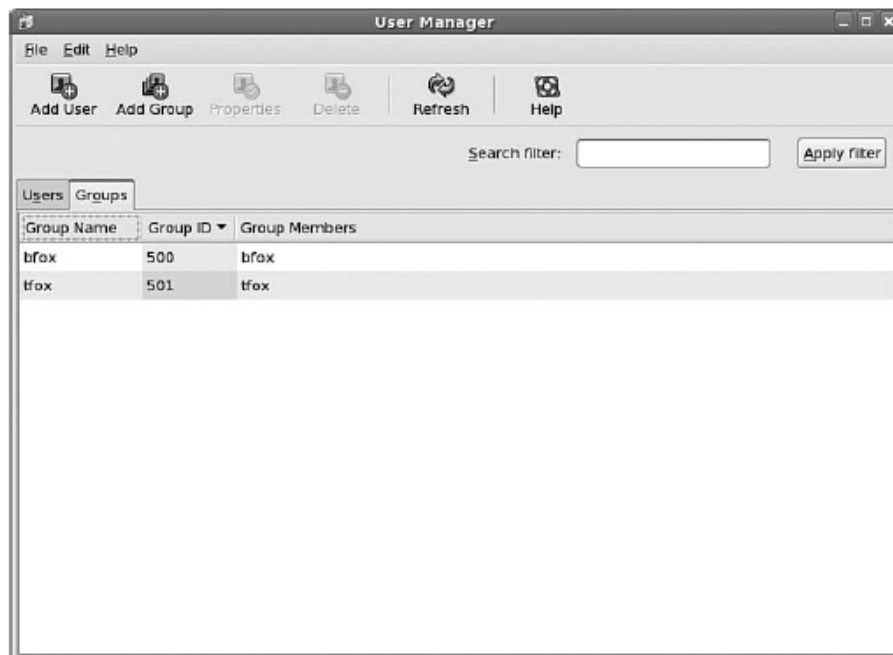
- User Name: [text input]
- Full Name: [text input]
- Password: [text input]
- Confirm Password: [text input]
- Login Shell: [dropdown menu showing /bin/bash]
- ☒ Create home directory
  - Home Directory: [text input showing /home/]
- ☒ Create a private group for the user
- ☐ Specify user ID manually
  - UID: [spin box showing 500]

Buttons: Cancel, OK

Figure 1.2: Adding a New User

## 1.2.4 Managing Groups

As discussed earlier, every user is assigned to a default private group with the same name as the user. But the user can also become a member of other groups. The graphical program to manage the groups can be invoked by selecting the **Users and Groups** option from the **Administration** menu. The **User Manager** window, with the existing groups being listed in the **Groups** tab, is displayed as shown in figure 1.3.



The 'User Manager' window has a menu bar (File, Edit, Help) and a toolbar with icons for Add User, Add Group, Properties, Delete, Refresh, and Help. Below the toolbar is a search filter field and an 'Apply filter' button. The main area has two tabs: 'Users' and 'Groups'. The 'Groups' tab is active, displaying a table of existing groups.

Group Name	Group ID	Group Members
bfox	500	bfox
tfax	501	tfax

Figure 1.3: List of Existing Groups

A new group can be added to the system using the **Add Group** button in the **User Manager** window. This invokes the **Create New Group** window shown in Figure 1.4. The details of the new group such as the name of the group, to be created and the group ID, can be specified here. If the user wants to assign a group ID manually, the checkbox **Specify group ID manually** needs to be checked. When the check box is checked, the user can select a group ID from the list box.





Figure 1.4: Adding a Group

## 1.3 Basic Linux Commands

Linux is a command line operating system. The basic commands in Linux deal with the manipulation of files and directories. The commands in Linux are case-sensitive.

The general syntax for a command is:

### Syntax:

```
command [options]
```

The commands are usually in lowercase letters. To execute a command without giving any options or arguments, the command is typed at the shell prompt. Most commands have various options that can be specified along with the command; these options control how the output is displayed.

### 1.3.1 The date Command

This command is used to display the current system date and time.

### Syntax:

```
[student@localhost ~]$ date [options]
```

The common options that can be used with the `date` command are summarized in table 1.1.

Option	Function
%m	Displays month of the year (in digits)
%d	Displays day of the month (in digits)
%y	Displays year (last two digits)
%D	Displays date as mm/dd/yy
%H	Displays hour (00 to 23)
%M	Displays minutes (00 to 59)
%S	Displays seconds (00 to 59)
%T	Displays time as HH:MM:SS

Table 1.1: Options of the date command

The code in Code Snippet 1 demonstrates the use of the `date` command with different options.

### Code Snippet 1:

```
[student@localhost ~]$ date "+%T"
21:36:42

[student@localhost ~]$ date "+%y"
08
```

The `date` command can be used by the system administrator only, to change the system date and time.

### 1.3.2 The who Command

This command is used to display the names of all the users currently logged on to the system.

#### Syntax:

```
[student@localhost ~]$ who [options]
```

The common options available with the `who` command are summarized in table 1.2.

Option	Function
-b	Indicates the most recent startup time and date
-l	Lists any login process
-H	Displays a header
-q	Prints only the login names and the number of users logged in

**Table 1.2: Options of the who command**

The code in Code Snippet 2 demonstrates the use of the `who` command.

#### Code Snippet 2:

```
[student@localhost ~]$ who
student pts/0 2008-03-26 17:36 (:0.0)
```

The `pts` in the output denotes a remote terminal. A remote terminal is a terminal connected from a machine other than the server.

The output of the `who` command contains four columns as explained in table 1.3.

Column#	Description
1	Displays a login name
2	Displays a terminal type and number
3	Specifies date and time of logging in
4	Specifies the remote host name of the terminal of the users who have not logged in to the server

**Table 1.3: Output of the who command**

The code in Code Snippet 3 prints the login names and the total number of users currently logged on to the system.

#### Code Snippet 3:

```
[student@localhost ~]$ who -q
student
# users=1
```

### 1.3.3 The man Command

This command is used to display pages from the Linux reference manual that is installed along with the Linux operating system.

#### Syntax:

```
[student@localhost ~]$ man [-] [-k keywords] topic
```

The arguments to the `man` command are summarized in table 1.4.

Argument	Function
-	Displays content of the manual without pausing
-k keywords	Searches for the keywords specified within the available manuals
topic	Displays the manual for the command typed in

**Table 1.4: Arguments of the man command**

The code in Code Snippet 4 lists the help information pertaining to the `ls` command.

**Code Snippet 4:**

```
[student@localhost ~]$ man ls
```

**1.3.4 Reading directory and Files**

Some of the commands for reading files and directories are explained in detail in this section.

**➤ The ls Command**

This command displays the names of files and sub-directories within a directory.

**Syntax:**

```
[student@localhost ~]$ ls [options]
```

The common options available with the `ls` command are summarized in table 1.5.

Options	Function
-a	Lists all the files, including hidden files
-F	Displays the file type along with the name
-R	Gives a recursive listing. In other words, displays the contents of the specified directory and sub directories
-r	Displays files and sub-directories in the reverse order
-S	Lists all files sorted by file size
-A	Displays the hidden files as well as the files beginning with '.'
-l	Displays a detailed list of files and directories

**Table 1.5: Options of the ls command**

The code in Code Snippet 5 lists all the files and subdirectories belonging to the student directory.

**Code Snippet 5:**

```
[student@localhost ~]$ ls /home/student
Desktop linuxtest.txt
```

The output of the `ls` command displays the filenames but not the file types. The code in Code Snippet 6 displays a detailed list of files and directories using the `-l` option with the `ls` command.

**Code Snippet 6:**

```
[student@localhost ~]$ ls -l
total 2
-rw-rw-r-- 1 student student 134 Jun 21 00.18 Desktop
drwxr-xr-x 1 student student 10 Jun 21 13.18 x
```

Every file or folder in Linux has access permissions. The three types of permissions are as follows:

- ✧ **read permission (r):** It means that the content of the file can be read using the `cat` command. For directories, it means that the content of the directory can be listed with the `ls` command. The octal value assigned to 'read' permission is 4.
- ✧ **write permission (w):** It means that the file can be modified and saved. For directories, it means that files can be created in that directory by the user. The octal value assigned to 'write' permission is 2.
- ✧ **execute permission (x):** It means that the file can be executed by the shell when its name is typed at the command prompt. For directories, it means that the user has the ability to traverse its tree in order to access files or subdirectories. The files inside the directory cannot be seen unless the read permission is set. The octal value assigned to 'execute' permission is 1.

Consider the following permission block: `drw-r---x`

The first character indicates that it is a directory. The next nine characters are permissions. The first three of the nine characters stands for the permissions for the Owner, while the next three depict the permissions for the Group and the last three characters stand for the permissions for Others. Table 1.6 describes the various columns in the output of the `ls -l` command:

Column#	Description
1	Specifies file type and File Access Permissions (FAP)
2	Specifies symbolic links, that is a file that points to another file
3	Specifies the name of the file owner
4	Specifies the name of the group owner
5	Specifies the file size in bytes
6, 7, and 8	Specifies the last file modification date and time
9	Specifies the file name

**Table 1.6: Output of ls-l command**

The code in Code Snippet 7 displays the file type and the file name of all the files (including the hidden files) in the student directory.

#### Code Snippet 7:

```
[student@localhost ~]$ ls -a -F /home/student
.bashhistory      .gnome
.bashlogout       .gnome2
.bash_profile     linuxtest.txt
Desktop/          .thumbnails
```

#### ➤ The dir and vdir Commands

The `dir` command works like the default `ls` command, as it lists the files in a sorted manner. The code in Code Snippet 8 demonstrates the use of the `dir` command.

#### Code Snippet 8:

```
[student@localhost ~]$ dir
News          axhome       nsmail       search
author.msg    documents    reading      vultures.msg
auto          mail         research
```

The `vdir` command works like the `ls -l` option and presents a long listing by default. The code in Code Snippet 9 demonstrates the use of the `vdir` command.

#### Code Snippet 9:

```
[student@localhost ~]$ vdir
total 10
drwxr-xr-x  2 bball  bball      1024 Nov 12 08:20 News
-rw-rw-r--  1 bball  bball      4766 Nov 12 07:41 author.msg
drwxrwxr-x  2 bball  bball      1024 Nov  5 10:04 auto
drwxrwxr-x  3 bball  bball      1024 Nov 12 13:54 axhome
drwxrwxr-x  2 bball  bball      1024 Nov 12 15:13 documents
drwx----- 2 bball  bball      1024 Nov 12 14:02 mail
drwx----- 2 bball  bball      1024 Sep 15 01:57 nsmail
drwxrwxr-x  2 bball  bball      1024 Oct 29 20:28 reading
drwxrwxr-x  5 bball  bball      1024 Nov  5 10:03 research
-rwxrwxr-x  1 bball  bball       200 Oct 24 13:24 search
```

#### ➤ The tree command

This command is used to list the contents of the directory in a tree like format. This helps in knowing how the directories are related to each other.

##### Syntax:

```
[student@localhost ~]$ tree [options]
```

The common options available with the `tree` command are summarized in table 1.7.

Options	Function
-a	Displays all files, except the hidden files
-d	Lists the directories only
-f	Displays the full path prefix for each file
-p	Displays the file permissions for each file
-s	Displays the size of each file along with the name
-r	Sorts the output in reverse alphabetical order
-L level	Specifies the maximum display depth of the directory tree

**Table 1.7 Options of the tree command**

The code in Code Snippet 10 demonstrates the use of the `tree` command.

**Code Snippet 10:**

```
[student@localhost ~]$ tree
.
|-- projects
|   |-- current
|   `-- old
|       |-- 1
|       `-- 2
`-- trip
    `-- schedule.txt

4 directories, 3 files
[student@localhost ~]$
```

The code in Code Snippet 11 displays the first level of sub-directories.

**Code Snippet 11:**

```
[student@localhost ~]$ tree -L 1
.
|-- Desktop
|-- progs
|-- temp
3 directories 0 files
[student@localhost ~]$
```

➤ **The touch Command**

This command is used for two purposes. First, to create a file and second, to update the file's modification date. It is available in GNU file utilities package.

**Syntax:**

```
[student@localhost ~]$ touch [Options] { File(s) }
```

The common options available with the `touch` command are summarized in table 1.8.

Option	Function
-a	Changes the access time of the file specified. It does not change the modification time unless -m is also specified.
-c	Does not create a new file if the file already exists.
-f	Attempts to force the execution of touch even if there are read and write restrictions on a file.
-m	Changes only the modification time
-r file	Uses the access and modification times of file

**Table 1.8: Options of the touch command**

The code in Code Snippet 12 creates a file named "linuxbasics.txt", if the file does not already exist. If the file already exists, the accessed / modification time is updated for the file linuxbasics.txt.

**Code Snippet 12:**

```
[student@localhost ~]$ touch linuxbasics.txt
```

A new file named linuxbasics.txt is created.

## ➤ The cat Command

This command is used to display the contents of a file on the screen or store it in another file. This command is also used to create, combine, overwrite, or append files.

### Syntax:

```
[student@localhost ~]$ cat filename [options]
```

The common options available with the `cat` command are summarized in table 1.9.

Options	Function
-n	Precedes each line with a line number
-u	Does not buffer the output
-e	Displays a character \$ at the end of each line
-b	Omits line numbers from blank lines
-t	Displays tabs in the output

**Table 1.9: Options of the cat command**

The code in Code Snippet 13 reads the files `file1.txt` and `file2.txt`, and adds the contents of these files into a third file-`file3.txt`. Suppose `file1.txt` contains:

cartoons are good

especially toons like tom (cat)

hello world!

and `file2.txt` contains:

I too

the number one song

they love us

### Code Snippet 13:

```
[student@localhost ~]$ cat file1.txt file2.txt > file3.txt
[student@localhost ~]$ cat file3.txt
cartoons are good
especially toons like tom (cat)
hello world!
I too
the number one song
they love us
```

The code in Code Snippet 14 reads the files `file1.txt` and `file2.txt`, combines these two files into a `file3.txt` file. It precedes each line in the output file with a line number, and appends the character `$` at the end of each line.

### Code Snippet 14:

```
[student@localhost ~]$ cat -n -e file1.txt file2.txt > file3.txt
[student@localhost ~]$ cat file3.txt
1 cartoons are good $
4 I too $
5 the number one song $
6 they love us $
```



### ➤ The more Command

This command is used to display the contents of a large file in a single screen. There are several options provided along with the `more` command to navigate through the file.

#### Syntax:

```
[student@localhost ~]$ more [options]
```

The common options available with the `more` command are summarized in table 1.10.

Options	Function
-c	Clears and redraws the screen before displaying
-d	Displays error message if an unknown command is used
-i	Performs case insensitive pattern matching in searches
+num	Specifies the starting line number
-u	Ignores backspace and underscores

**Table 1.10: Options of the more command**

The code in Code Snippet 15, uses the `more` command to display the contents of the file `myfile.txt` starting from line 3.

#### Code Snippet 15:

```
[student@localhost ~]$ more +3 myfile.txt
```

### ➤ The exit Command

After logging on to the system, the Linux session will continue until the user instructs the Linux shell to terminate the session. To terminate the Linux session, the `exit` command is used. The system then displays the **login:** prompt on the screen.

### ➤ The shutdown Command

The `shutdown` command is used to shutdown the Linux operating system. The common options available with the `shutdown` command are summarized in table 1.11.

#### Syntax:

```
[student@localhost ~]$ shutdown [options]
```

Options	Function
-h	Halts after shutdown
-r	Reboots after shutdown
-c	Cancels the shutdown

**Table 1.11: Options of the shutdown command**

## 1.4 Advanced Linux Commands

Some of the commands related to processes, disk usage and system information are explained in detail in this section.

### 1.4.1 The df Command

The `df` command stands for Disk Free. This command is used to display the amount of space used and the amount of free disk space available on the currently mounted filesystems.

#### Syntax:

```
[student@localhost ~]$ df [option] [File_name]
```

When the `df` command is invoked without using any arguments, it displays the used and free disk space in blocks. The common options available with the `df` command are summarized in table 1.12.

Options	Function
-h	Displays sizes in megabytes and gigabytes and appends the data with M and G respectively.
-k	Displays the data in 1KB blocks
-i	Displays the inode usage
-T	Adds type of each filesystem to the report
-help	Displays a brief help message

Table 1.12: Options of the df command

The code in Code Snippet 16 displays the sizes in an easy to read format.

#### Code Snippet 16:

```
[student@localhost ~]$ df -h

Filesystem      Size  Used Avail Use% Mounted on
/dev/hda2       28G   7.6G   19G   29% /
tmpfs           252M    0   252M    0% /dev/shm
/dev/hda1       464M   37M   403M    9% /boot
/dev/hda3       8.3G  429M   7.5G    6% /var
nfs6:/home     520G  461G   60G   89% /home
```

### 1.4.2 The ps Command

The `ps`, or Process Status, command displays the processes that are currently running on the Linux system. A process is a running instance of a program. The function of `ps` command is similar to the “Task Manager” popup in Windows, which is obtained by pressing the **Ctrl+Alt+Del** keys.

#### Syntax:

```
[student@localhost ~]$ ps [option]
```

The common options available with the `ps` command are summarized in table 1.13.

Options	Function
-f	Generates a full listing of all the processes
-A	Displays information of all processes
-a	Displays information about all frequently requested processes
-e	Displays information about every process that is currently running

Table 1.13: Options of the ps command

The code in Code Snippet 17 lists the currently running processes.

**Code Snippet 17:**

```
[student@localhost ~]$ ps
PID TTY          TIME CMD
3511 pts/1        00:00:00 bash
3514 pts/1        00:00:00 ps
```

The `ps` command displays the process ID, the terminal associated with the process, the CPU time and the executable shell name. The code in Code Snippet 18 displays all the information related to frequently requested processes.

**Code Snippet 18:**

```
[student@localhost ~]$ ps -f -a
UID      PID     PPID    C  STIME   TTY      TIME    CMD
student  2840    2809    0  16:16   pts/0    00:00:00 ps -f -a
```

**1.4.3 The kill Command**

The system often requires to communicate the occurrence of an event to a process. This is done by sending a signal to the process. The signals are sent using the signal name or signal number. The name and purpose of some signals are as follows:

- Signal 15, TERM (default) - Terminate cleanly
- Signal 9, KILL - Terminate immediately
- Signal 1, HUP - Re-read configuration files
- `man 7 signal` shows complete list

The `kill` command is used with the `ps` command. The command will stop execution of one process.

**Syntax:**

```
[student@localhost ~]$ kill [options] [pids]
```

If the signal is not specified, a **TERM** signal is sent, which is used to kill the processes that do not catch the signal.

The common options available with the `kill` command are summarized in table 1.14.

Options	Function
-a	Kills all the processes having the name specified
-l	Lists all signals
-p	Prints the process ID of the named processes without sending the signal
-s	Specifies the signal number or name of the signal

Table 1.14: Options of the kill command

## Working with Red Hat Enterprise Linux 5.0

The `kill -9` signal forces the process to die. The `pids` attribute specifies the list of processes to which the `kill` command sends a signal.

### Code Snippet 19:

```
[student@localhost ~]$ kill -s kill 100 -165
```

The code in Code Snippet 19, sends the **SIGKILL** signal to the process whose process ID is 100 and to all processes whose process group ID is 165.

### 1.4.4 The `uname` Command

This command is used to displays the system information.

#### Syntax:

```
[student@localhost ~]$ uname [options]
```

The common options available with the `uname` command are summarized in table 1.15.

Options	Function
-a	Prints all the basic information currently available from the system
-r	Prints the operating system release level
-s	Prints the operating system name
-m	Prints the machine hardware type
-p	Prints the machine's processor type
-v	Prints the operating system version

Table 1.15: Options of the `uname` command

### Code Snippet 20:

```
[student@localhost ~]$ uname -arv
Linux localhost.localhostdomain 2.6.18-8.el5 #1 SMP Fri Jan 26 14:14:15 EST 2007
i686 i686 i286 GNU/Linux
[student@localhost ~]$
```

The code in Code Snippet 20 prints the basic information of the system along with the OS release level and the OS version.

### 1.4.5 The `tty` command

As Linux treats even the terminals as files, the names of these terminals can be displayed using the `tty` (teletype) command.

#### Syntax:

```
[student@localhost ~]$ tty
/dev/pts/0
```

The terminal filename is **tty01** and is resident in the **/dev** directory.

### 1.4.6 Printing Commands

After creating a document, the user might wish to print it. Linux provides various commands for printing a document. Each printer is associated with one or more queues. The documents to be printed are sent to the queue, and not to the printer directly. Different queues for the same printer may have different priorities. The task of setting up the print queues is performed by the system administrator. Once the document reaches the queue for printing, it is called a job.

Some of the printing commands are as follows:

### ➤ The **lpr** command

This command sends the job to the print queue.

#### Syntax:

```
[student@localhost ~]$ lpr [options] filename
```

The common options available with the **lpr** command are summarized in table 1.16.

Options	Function
-P destination	Name of the printer on which to print
-# number	Prints the specified number of copies
-T title	Prints a title on the banner page of the output
-q	Holds a job for printing

**Table 1.16: Options of the lpr command**

#### Code Snippet 21:

```
[student@localhost ~]$ lpr -P accounting -#5 report.txt
```

The code in Code Snippet 21 prints 5 copies of the file `report.txt` on a printer named `accounting`.

### ➤ The **lpq** command

This command is used to check the print spool queue for the status of the print jobs. It displays username, position in the queue, filenames, job number, and total file size (in bytes) for each job.

#### Syntax:

```
[student@localhost ~]$ lpq [option] printer
```

The common options available with the **lpq** command are summarized in table 1.17.

Options	Function
-P destination	Displays information about the printer or class of printers
-a	Reports jobs on all printers
-U username	Specifies an alternate username for the user
-E	Uses encryption to connect to a print server

**Table 1.17: Options of the lpq command**

#### Code Snippet 22:

```
[student@localhost ~]$ lpq -P lp0
```

The code in Code Snippet 22 displays the information about the printer `lp0`.

### ➤ The **lprm** command

This command is used to remove a job from a queue.

#### Syntax:

```
[student@localhost ~]$ lprm [options]
```

The common options available with the **lprm** command are summarized in table 1.18.

Options	Function
-P destination	Displays information about printer or class of printers
-h server[:port]	Specifies an alternate server
-U username	Specifies an alternate username

Table 1.18: Options of the lprm command

**Code Snippet 23:**

```
[student@localhost ~]$ lprm -P kill 385
```

The code in Code Snippet 23, removes request ID 385 from destination.

**1.4.7 Running a process in the background**

A background process is a child of the process that spawned it. The parent processes, however, does not wait for the child process to complete. When a process is started in the background, a new bash sub shell is created. The & is the shell's operator used to run a process in the background. To start a process in the background, terminate the process with an & symbol.

**Code Snippet 24:**

```
[student@localhost ~]$ sort file1.txt file2.txt &
[1] 2975
```

The shell immediately returns a number which is the PID of the invoked command. The prompt is returned, and the shell is ready to accept another command, even though the previous command has not been terminated.

Background execution of a job is a useful feature that allows the user to execute the less important jobs in the background, while running more important ones in the foreground.

## The Session In Brief

- The X Window System is a Graphical User Interface in Linux, which provides the foundation for Window Managers such as GNOME and KDE. The desktop environment is a user interface that runs on these Window Managers.
- When a new user is added to the system, a private user group is automatically created with the same name as the user's name.
- The general syntax for Linux commands is `command - [options] [argument]`.
- The root user is a special administrative account, that has unrestricted access to all the files, devices, and programs in the system.
- The basic Linux commands include:
  - ✧ The date command
  - ✧ The who command
  - ✧ The man command
  - ✧ The dir, vdir, tree, cat, and more commands for reading directories and files
  - ✧ The exit command
  - ✧ The shutdown command
- The advanced Linux commands include:
  - ✧ The df Command
  - ✧ The ps Command
  - ✧ The kill Command
  - ✧ The uname Command
  - ✧ The tty Command
  - ✧ The printing commands like lpr, lpq, lprm
- The & is the shell's operator used to run a process in the background. To stop a background process, the command must be terminated with the & symbol.



## Check Your Progress

1. To display all the files, including the hidden files in the current directory, the \_\_\_\_\_ option of the `ls` command must be used.
  - a. -A
  - b. -a
  - c. -l
  - d. -F
2. To display the date in the mm/dd/yy format, the \_\_\_\_\_ option of the `date` command must be used.
  - a. %m
  - b. %d
  - c. %D
  - d. %y
3. When a new user is added to the system, a \_\_\_\_\_ is automatically formed.
  - a. public user group
  - b. private user group
  - c. duplicate user group
  - d. secure user group
4. The \_\_\_\_\_ command is used to display the currently running processes in the Linux system.
  - a. df
  - b. ps
  - c. show
  - d. ls
5. The \_\_\_\_\_ command is used to combine two or more files.
  - a. cat
  - b. combine
  - c. ls
  - d. more

## Working with the X Window System (Lab)

### Session Objectives:

*At the end of this session, you will be able to -*

- Perform the Red Hat Enterprise 5.0 installation
- Create users and log into the system
- Use Linux commands

### Part I - 60 Minutes

---

#### Exercise 1: Perform the Red Hat Enterprise 5.0 installation

Before beginning the installation, the user must configure the BIOS settings to boot from the CD-ROM.

1. **Switch on the system.**
2. **Press <Delete> key for entering the BIOS settings.**
3. **Change the boot order so that the system searches the CD-ROM drive first for any boot media.**
4. **Save the changes and exit.**

The user can begin the installation of Red Hat Enterprise Linux 5.0 from compact discs available from any Linux distributor.

To start the installation:

5. **Insert the first CD of Red Hat Enterprise Linux 5.0 in the CD-ROM drive.**
6. **Reboot the system.**

A welcome screen appears as shown in figure 2.1.



Figure 2.1: Welcome Screen

**7. Click Next.**

The **Language Selection** screen appears as shown in figure 2.2. To view the Release notes, you can click the Release Notes button. The Release Notes is a summary of recent changes, enhancements and bug fixes in a particular software release.



Figure 2.2: Language Selection Screen

**8. Select English and click Next.**

The **Keyboard Configuration** screen appears as shown in figure 2.3.



Figure 2.3: Keyboard Configuration Screen

**9. Select U.S. English and click Next.**

The **Installation Number** dialog box appears as shown in figure 2.4.



Figure 2.4: Installation Number dialog box

10. Enter the installation number and click OK.
11. Click Next.

The **Disk Partitioning** screen appears as shown in figure 2.5. The installation number entered determines the package selection set that will be used by the installer.

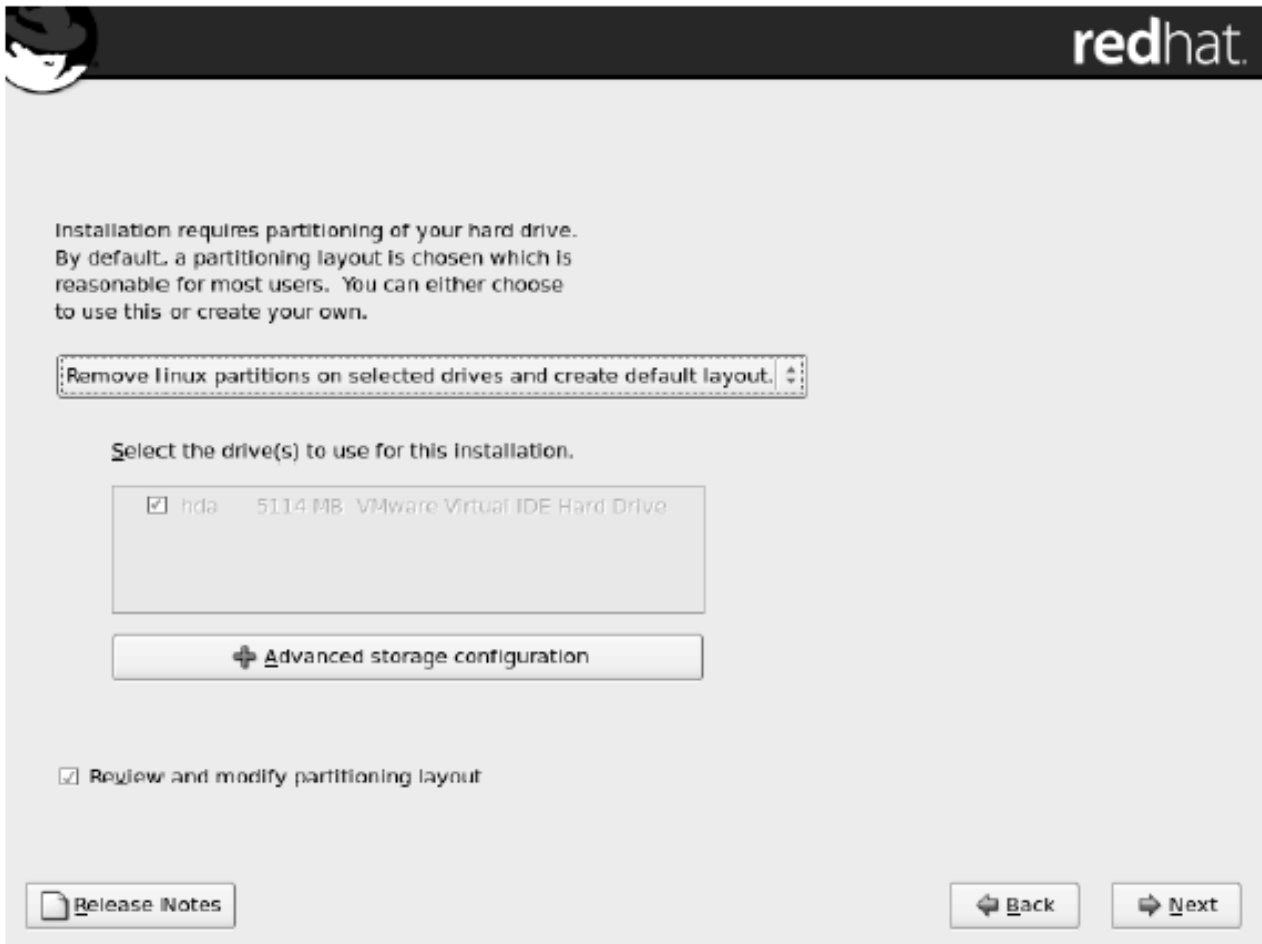


Figure 2.5: Disk Partitioning Screen

**12. Select “Remove linux partitions on selected drives and create default layout”, and click Next.**

The partitions created in **Disk Druid** appear as shown in figure 2.6. The “Remove linux partitions on selected drives and create default layout” option removes Linux partitions created from the previous installations. The other partitions in the system remain unaffected.



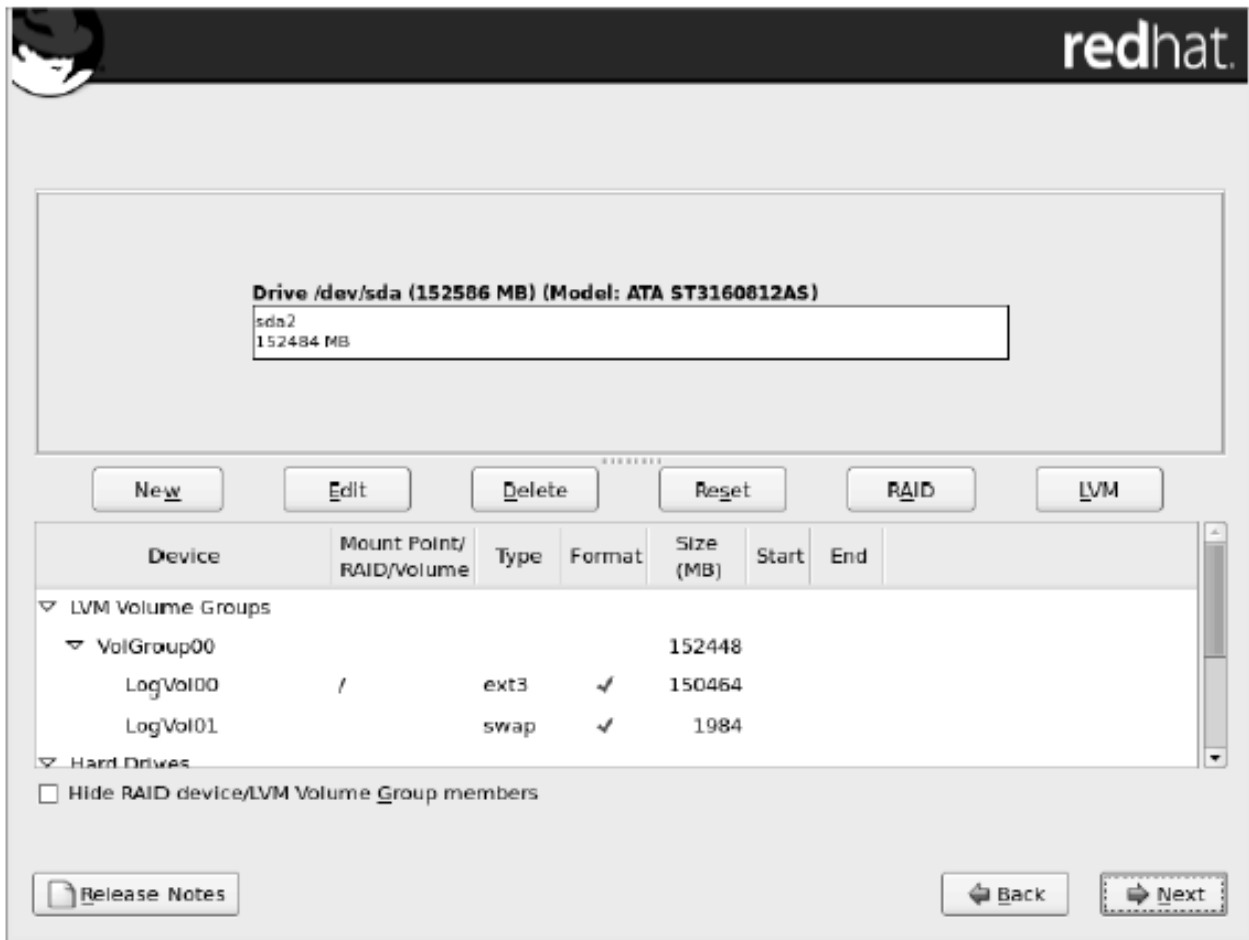
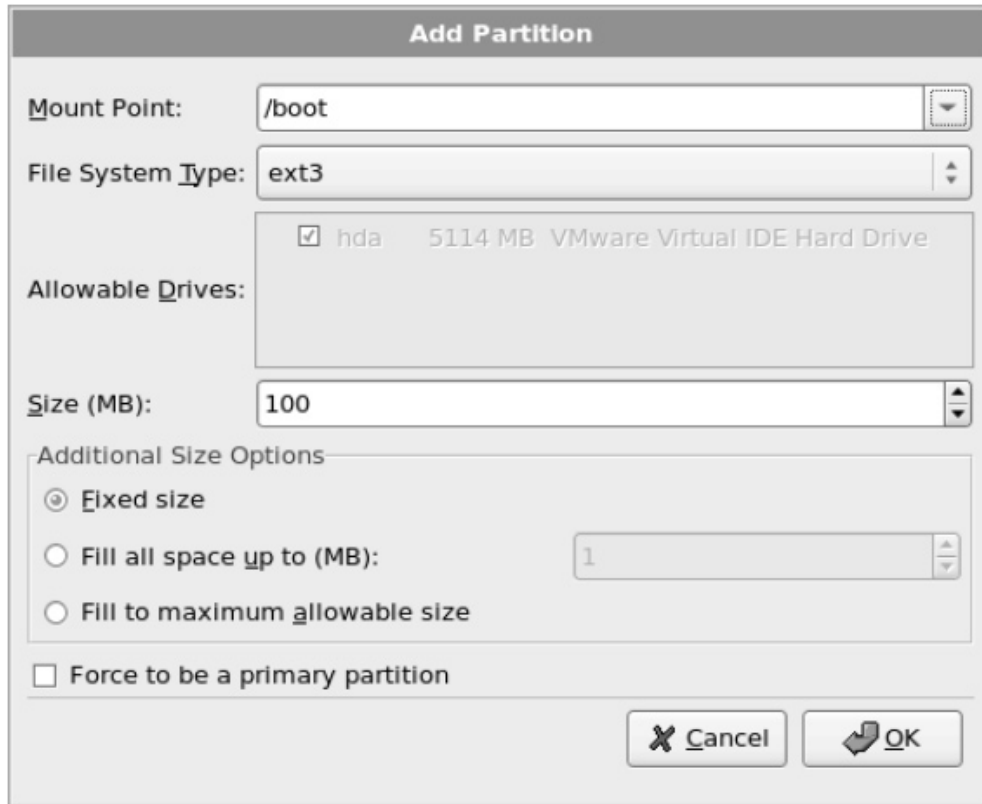


Figure 2.6: Partitioning with Disk Druid

13. Click New.

The **Add Partition** screen appears as shown in figure 2.7.

14. Specify the mount point as `/boot`.



The 'Add Partition' dialog box is shown with the following settings:


- Mount Point:** /boot
- File System Type:** ext3
- Allowable Drives:** A list containing one entry: ☒ hda 5114 MB VMware Virtual IDE Hard Drive
- Size (MB):** 100
- Additional Size Options:**
  - ☒ Fixed size
  - ☐ Fill all space up to (MB): 1
  - ☐ Fill to maximum allowable size
- ☐ Force to be a primary partition

Buttons at the bottom: Cancel, OK

Figure 2.7: Add Partition Screen

**15. Click OK.**

The **Boot Loader Configuration** screen appears as shown in figure 2.8. The boot loader is required to boot the system without any boot media. It is the first program that runs when the system starts.



☒ The GRUB boot loader will be installed on /dev/hda.  
☐ No boot loader will be installed.

You can configure the boot loader to boot other operating systems. It will allow you to select an operating system to boot from the list. To add additional operating systems, which are not automatically detected, click 'Add.' To change the operating system booted by default, select 'Default' by the desired operating system.

Default	Label	Device
<input checked="" type="checkbox"/>	Red Hat Enterprise Linux Server	/dev/VolGroup00/LogVol00

[Add](#)  
[Edit](#)  
[Delete](#)

A boot loader password prevents users from changing options passed to the kernel. For greater system security, it is recommended that you set a password.

☐ Use a boot loader password [Change password](#)

☒ Configure advanced boot loader options

[Release Notes](#)
[Back](#)
[Next](#)

Figure 2.8: Boot Loader Configuration

**16. Select “Configure advanced boot loader” options check box and click Next.**

The **Advanced Boot Loader** screen appears as shown in figure 2.9. The options such as changing the drive order or passing options to the kernel can be configured after selecting the check box “Configure advanced boot loader options”.



Figure 2.9: Advanced Boot Loader Screen

**17. Click Next.**

The **Network Configuration** screen, which displays the network devices that have been detected, appears as shown in figure 2.10.



**RED HAT ENTERPRISE LINUX 5**

**Network Devices**

Active on Boot	Device	IPv4/Netmask	IPv6/Prefix
<input checked="" type="checkbox"/>	eth0	DHCP	DHCP

[Edit](#)

**Hostname**

Set the hostname:

☒ automatically via DHCP

☐ manually  (e.g., host.domain.com)

**Miscellaneous Settings**

Gateway:

Primary DNS:

Secondary DNS:

[Release Notes](#) [Back](#) [Next](#)

Figure 2.10: Network Configuration Screen

#### 18. Select the network device and click Edit.

The **Edit Interface eth0** screen appears as shown in figure 2.11. The **Edit** option in the **Network Configuration** screen displays a dialog box for editing the IP address and the netmask for the selected device.

**Edit Interface eth0**

**Configure eth0 - Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]**

Hardware address: 00:0C:29:42:45:96

☒ Use dynamic IP configuration (DHCP)

☒ Enable IPv4 support

☒ Enable IPv6 support

☒ Activate on boot

	Address	Prefix (Netmask)
IPv4:	<input type="text"/>	<input type="text"/>
IPv6:	<input type="text"/>	<input type="text"/>

Figure 2.11: Editing a Network Device

The IP address and Netmask can be specified here.

19. Click OK to return the control to the Network Configuration Screen.
20. Click Next.

The **Time Zone Configuration** screen appears as shown in figure 2.12.

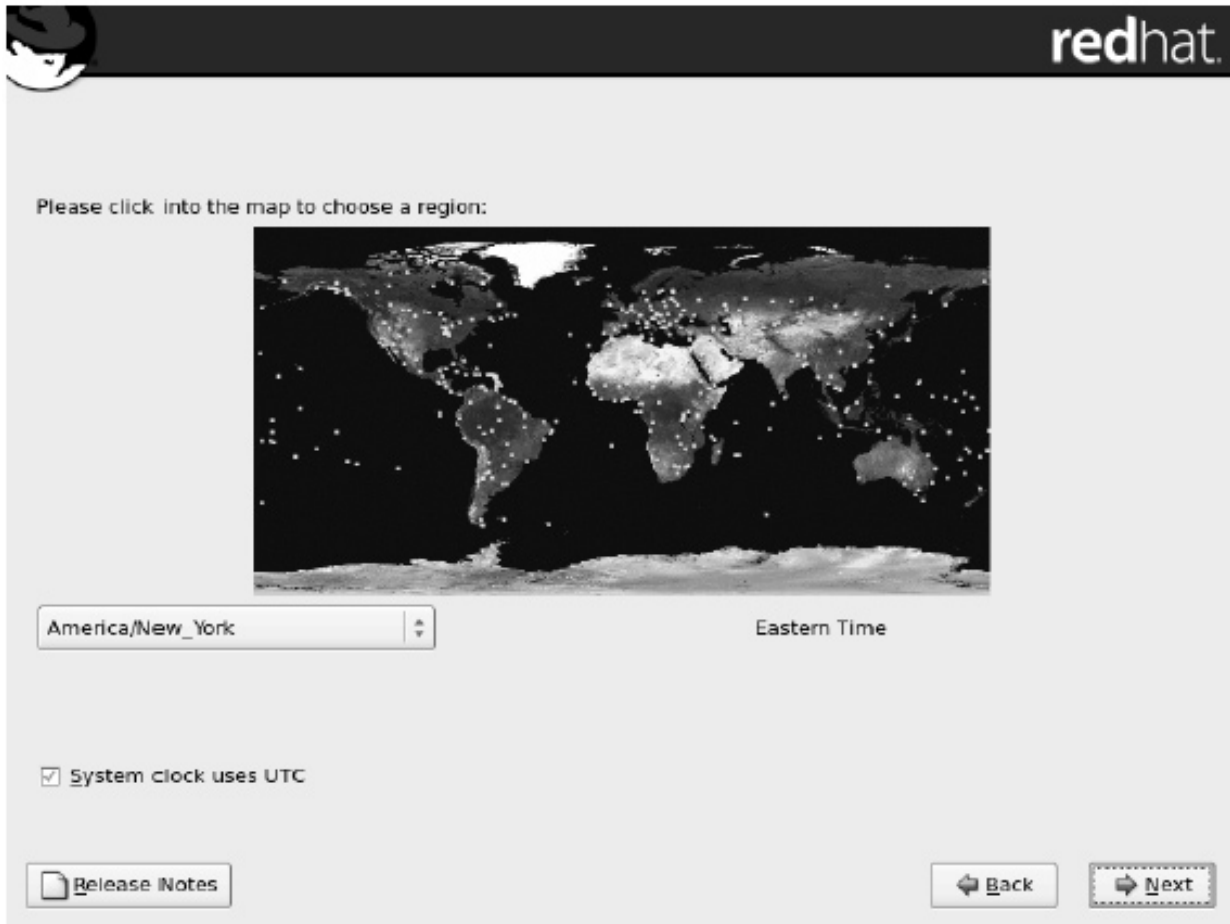


Figure 2.12: Time Zone Selection Screen

21. Set the time zone for the system.
22. Select “System clock uses UTC” check box if the system clock is based on Universal Time Coordinated time.

Universal Time Coordinated is International Atomic Time (TAI) with leap seconds added at irregular intervals to compensate for the Earth’s slowing rotation.

23. Click Next.

The **Root Password** screen appears as shown in figure 2.13.



Figure 2.13: Root Password Screen

24. Type the password for the root account.

25. Click Next.

The **Package Installation Default** screen appears and the details of the default package set for Red Hat Enterprise Linux installation are displayed.

26. Click Next.

A screen preparing for the installation of Red Hat Enterprise Linux appears.

27. Click Next.

The **Installing Packages** screen appears.

After the installation is complete, the Installation Complete screen appears as shown in figure 2.14.





Figure 2.14: Installation Complete Screen

**28. Click Reboot.**

After rebooting, the **Setup Agent** welcome screen appears as shown in figure 2.15.



Figure 2.15: Setup Agent Screen

**29. Click Forward.****30. Agree to the license agreement and click Next.**

## Working with Red Hat Enterprise Linux 5.0

The Date and Time screen appears.

31. Select current date from the Date area.
32. Enter current hours, minutes, and seconds in the Time area.
33. Click Next.

The **Create Profile** screen appears.

34. Create a profile and click Forward.

The Create User screen appears.

35. Enter the username, password, and click Forward.

The Sound Card screen appears.

36. Click the Play Test Sound button.

A sound plays as specified on the screen.

The last screen is the **Finish Setup** screen.

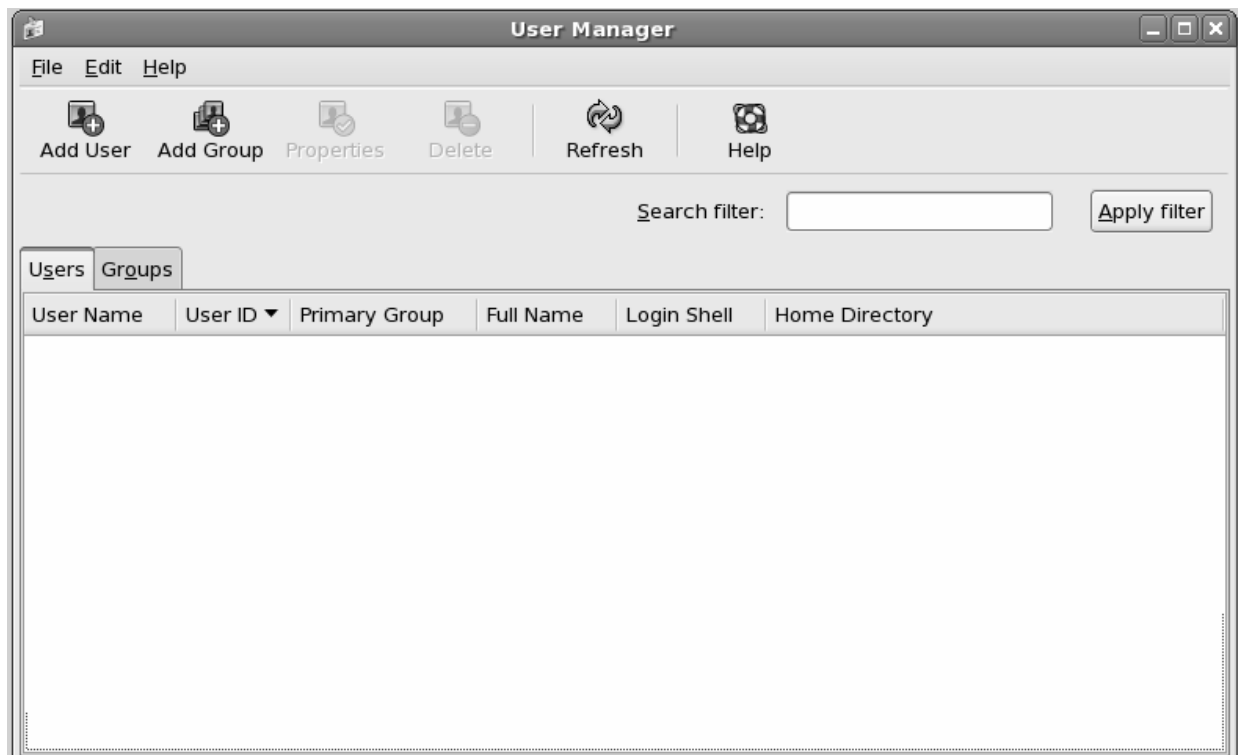
37. Click Next to exit the Setup Agent and go to the login screen.

The installation is now complete. The user is asked to enter the user name and password for logging on to the Linux system.

### Exercise 2: Creating a user and logging into the system

1. Login as root user, and to create a new user, click **System → Administration → Users and Groups**.

The **User Manager** Screen appears as shown in figure 2.16.



**Figure 2.16: User Manager**

2. Click Add User button to add a new user. Enter the details of the user as shown in figure 2.17.

The screenshot shows a 'Create New User' window with the following details:

- User Name:** student
- Full Name:** Aptech
- Password:** masked with asterisks
- Confirm Password:** masked with asterisks
- Login Shell:** /bin/bash
- ☒ **Create home directory**
- Home Directory:** /home/student
- ☒ **Create a private group for the user**
- ☐ **Specify user ID manually**
- UID:** 500
- Buttons:** Cancel, OK

Figure 2.17: Add User

3. Click OK.

The user student is created and the home directory `/home/student` is allocated as shown in figure 2.18. The **Add User** dialog box enables to add a new user by specifying the username, full name and password. The **Login Shell** option specifies the login shell for the user. The default login shell is `bash`. The **Create home directory** option allows you to specify whether a home directory is to be created for the user or not. The name and location of the home directory for the user can be specified in the **Home Directory** text box. The default home directory is `/home/username`. The **Create a private group for the user** option can be selected to have Linux create a private group with the same name as the user name. The **Specify user ID manually** enables to manually set the user id.

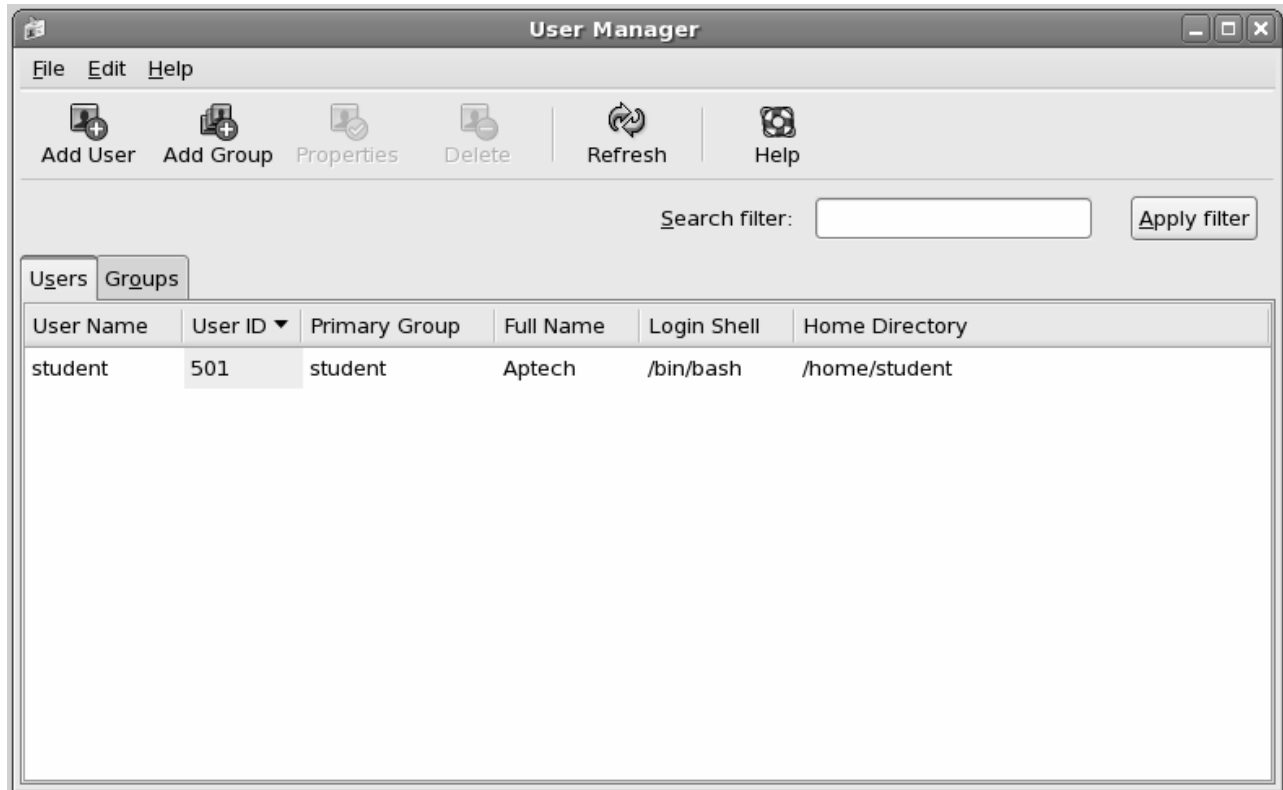


Figure 2.18: New User Details

**4. Click System → Log Out root.**

The root user is logged out. The user is prompted for the login again.

**5. Type student in Username box and password in Password box.**

The student is logged in and the graphical desktop screen appears as shown in figure 2.19.



Figure 2.19: Default Graphical Desktop

**6. Click on Applications → Accessories → Terminal.**

The user can use a GUI terminal window to get a basic text terminal without exiting the GUI.

### Exercise 3: Using Linux commands

#### Problem

Give the commands to perform the following activities:

- ✧ Display the current date and time of the system.
- ✧ Display the names of the users currently logged on to the system.
- ✧ Display the directories and sub-directories within the system.
- ✧ Display the status of the disk space usage on the system.
- ✧ Display the active processes for the current login session.

#### Analysis

The `date` command is used to print the system date and time. The `man` command provides documentation for the commands in Linux. The `who` command displays the list of all the users currently logged on to the system. The `ls` command is used to display the list of directories and sub-directories. The `tree` command displays the contents of the directory in a tree like format. The `df` command displays the amount of space used and the amount available on the currently mounted file system. The `ps` command displays the currently running processes in the Linux system. The `uname` command prints the system information.

#### Solution

##### 1. Type the following command to display the current date and time.

```
[student@localhost ~]$ date
Mon Feb 04 11:36:42 EST 2008
[student@localhost ~]$
```

##### 2. Type the following commands to display time and year.

```
[student@localhost ~]$ date +%T
11:36:59
[student@localhost ~]$ date +%y
08
[student@localhost ~]$
```

The “%T” option displays the time as HH:MM:SS. The “%y” option displays the last two digits of the year in a date.

##### 3. Type the following command to display the date in mm/dd/yy format along with the string “DATE :”.

```
[student@localhost ~]$ date "+DATE : %D"
DATE: 02/04/08
[student@localhost ~]$
```

##### 4. Type the following command to display the names of all the users currently logged on to the system.

```
[student@localhost ~]$ who
student pts/0      2008-03-26  17:36 (:0.0)
[student@localhost ~]$
```

The `ls` command displays the names of files and sub-directories within a directory.

5. Type the following command to display the names of the files and sub-directories within a directory.

```
[student@localhost ~]$ ls
Desktop  LinuxTutorials
[student@localhost ~]$
```

6. Type the following command to display a detailed list of files and directories.

```
[student@localhost ~]$ ls -l
total 16
drwxr-xr-x 4 student student 4096 Mar 28 16:37 Desktop
drwxrwxr-x 3 student student 4096 Mar 28 23:37 LinuxTutorials
[student@localhost ~]$
```

7. Type the following command to access the help related to the ls command.

```
[student@localhost ~]$ man ls
```

Figure 2.20 shows the output of the `man` command.

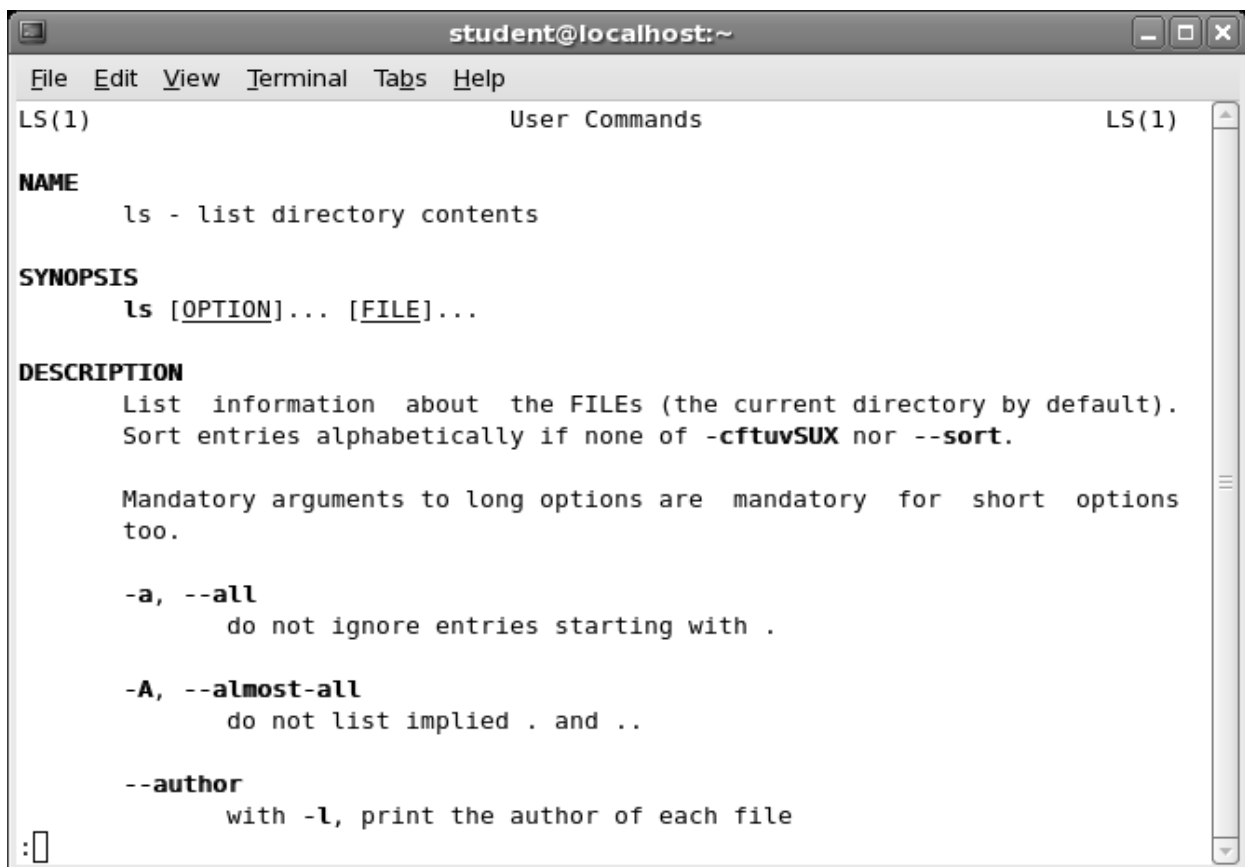


Figure 2.20: Output of `man ls` command

8. Type the following command to display the list of directories and sub directories.

```
[student@localhost ~]$ tree -d
.
|-- Desktop
'-- LinuxTutorials
    '--examples

3 directories
[student@localhost ~]$
```

The `-d` option of the `tree` command displays only the directories and not the files.

#### 9. Type the following command to display the status of the disk space usage of the system.

```
[student@localhost ~]$ df
Filesystem 1K-blocks    Used   Available  Use%  Mounted on
/dev/hda7   3968092  2982016    781252   80%  /
tmpfs       253384      0    253384    0%  /dev/shm
[student@localhost ~]$
```

The name of the root filesystem is `/dev/hda7`. The `tmpfs` is a Linux file system which keeps all files in virtual memory. The used space, free space, used percentage of the filesystem are also displayed. The path where the filesystem is mounted is also specified.

#### 10. Type the following command to display the status of the disk space usage of the system in easy readable format.

```
[student@localhost ~]$ df -h
Filesystem Size      Used   Available  Use%  Mounted on
/dev/hda7  3.8G      2.9G    763M      80%  /
tmpfs      248M        0    248M       0%  /dev/shm
[student@localhost ~]$
```

The `-h` option displays sizes in megabytes and gigabytes by appending the characters M and G respectively. Size 3.8G specifies that the size of `/dev/hda7` filesystem is 3.8 Gigabytes.

#### 11. Type the following command to display all the active processes for the current login session.

```
[student@localhost ~]$ ps
  PID   TTY      TIME    CMD
 2808   pts/0  00:00:00    bash
 2870   pts/0  00:00:00      ps
[student@localhost ~]$
```

The `ps` command displays the currently running processes in the Linux system.

#### 12. Type the following command to display the kernel and operating system version.

```
[student@localhost ~]$ uname -v -o
#1 SMP FRI JAN 26 14:15:21 EST 2007 GNU/Linux
[student@localhost ~]$
```

The `uname` command with the option `-o` displays the version of the operating system. The `-v` option displays the kernel version.

## Part II - 60 Minutes

---

1. Give the commands to do the following:
  - ✧ Create a new user named, davidblake.
  - ✧ Display the user's login name, terminal name, and write status along with the idle time, login time, office location and office number.
  - ✧ Display which user has logged on (appropriate headings have to appear over the columns).
  - ✧ Display the file type along with the name.
  - ✧ Display all the files in the reverse order.
  - ✧ Display all the hidden files.
  - ✧ Display the current working directory.
  - ✧ Display the machine's hardware type.



## Do It Yourself

1. Give the commands to:
  - ✧ Display the date in the following format  
"Mon FEB 04 10:10:10 IST 2008"
  - ✧ Display the date in the format  
"2008-02-13"
  - ✧ Display the date in the format  
"MM/DD/YY"
  - ✧ Display the time in the format  
"02:55:55 PM"
  - ✧ Display the time in the format  
"15:50"
  - ✧ Display the week number of the year
  - ✧ Display the Time Zone
2. Execute the commands to display the day of the week in the following formats:
  - a. Digit
  - b. Name in abbreviated form
  - c. Name in Full
3. Execute the commands to display files and directories from the /etc directory in a tree structure.
4. Execute the command to list only the directories from the /etc directory in a tree structure.

The background of the image is a grayscale, high-contrast photograph of a computer keyboard and a circuit board. The keyboard keys are prominent in the upper half, with labels like 'CTRL', 'SHIFT', and 'BACK TAB' visible. The lower half shows the intricate patterns of a circuit board, including various components and traces. The overall aesthetic is technical and modern.

“

**Nothing is a waste of time if you  
use the experience wisely**

”

# Working with the File System and Command Line

## Session Objectives:

At the end of this session, you will be able to -

- List the types of file systems
- Discuss the commands to manipulate files or directories
- Explain how to archive files
- Explain how to use the vi and vim editors
- Identify the commands to process strings

## 3.1 Types of file systems

Linux runs on almost every hardware platform, so it must support all the file system types supported by those hardware platforms. By using a common file system, other operating systems can share files with Linux when both are installed on the same computer.

A file system is the way in which the operating system organizes the data on the disk. A file system includes the methods the operating system uses to keep track of files on a disk or partition. File systems store the data in the form of data blocks. The data and metadata about the files is present in the file system. Metadata contains information such as the owner of the file, permissions assigned to the file and other such file related information.

Linux maintains file systems in a single hierarchical structure. The root directory is represented by a '/'. Linux sorts the directories in the descending order starting from the root directory.

The file system type is determined by how these data structures are defined, and how they are used. The file system type selected by the user determines the system performance, data reliability, crash recovery time, maximum number of files supported, and file access control specifications. Generally, the default file system types used in Linux are ext2fs, ext3fs, and ReiserFS.

Linux also allows the user to easily add new file system types. The user can download the driver for a specific file system from the appropriate Web site. The common file system types supported by Linux are summarized in table 3.1.

File system	Description
Extended File System -ext	It is an elaborate extension of the minix file system. It has been completely superseded by the second version of the extended file system.
Second Extended File System -ext2	It is the second extended file system designed as an extension of the extended file system, ext. ext2 is a high performance disk file system used by Linux, for fixed disks and removable media. ext2 offers the best performance in terms of speed and CPU usage, among all the other file systems supported by Linux.
Third Extended File System -ext3	It is an enhanced version of the ext2 file system. It offers the most complete set of journaling options available amongst journaling file systems. It is easy to switch back and forth between ext2 and ext3.

File system	Description
Reiser File System -reiserfs	It treats the entire disk partition as if it were a single database table. Directories, files, and file metadata are organized in an efficient data structure called the balanced tree. This file system differs from the traditional file systems in the way they operate, but it offers large speed improvements for any applications using many small files.
Minix File System -Minix	It is the file system used under the Minix operating system, and was the first file system type to be used under Linux. A 64MB partition size limit, short file names, and a single timestamp are the drawbacks of this file system. It is mostly useful for floppies and RAM disks.
FAT 16 bit -msdos	It is the file system used by DOS, Windows, and some OS/2 computers.
Virtual FAT File System -vfat	It is an extended DOS file system used with Microsoft Windows NT.
Network File System -NFS	It is used to access disks located on remote computers.
System V File System -sysv	It is an implementation of the System V/ Coherent file system for Linux.
NT File System -ntfs	It is an optional file system used on Windows NT, 2000, and XP operating systems. It provides improved performance and implements various security and administrative features. It logs activities, and recovers dynamically from hard disk crashes.
proc	It is a pseudo file system, which is used as an interface to kernel data structures, instead of reading and interpreting /dev/kmem. Files in this file system do not occupy any disk space.
smbfs	It is the network file system that supports the Server Message Block (SMB) protocol, used by Windows for workgroups, Windows NT, and LAN Manager. To use smbfs, the user requires a special mount program.
High Performance File System -hpfs	It is used in OS/2. This file system is read-only under Linux, due to lack of available documentation.

**Table 3.1: File System Types in Linux**

All the files in Linux are related to one another. The file system in Linux is a collection of all the related files organized in a hierarchical structure as shown in figure 3.1. It is organized in an inverted tree structure.

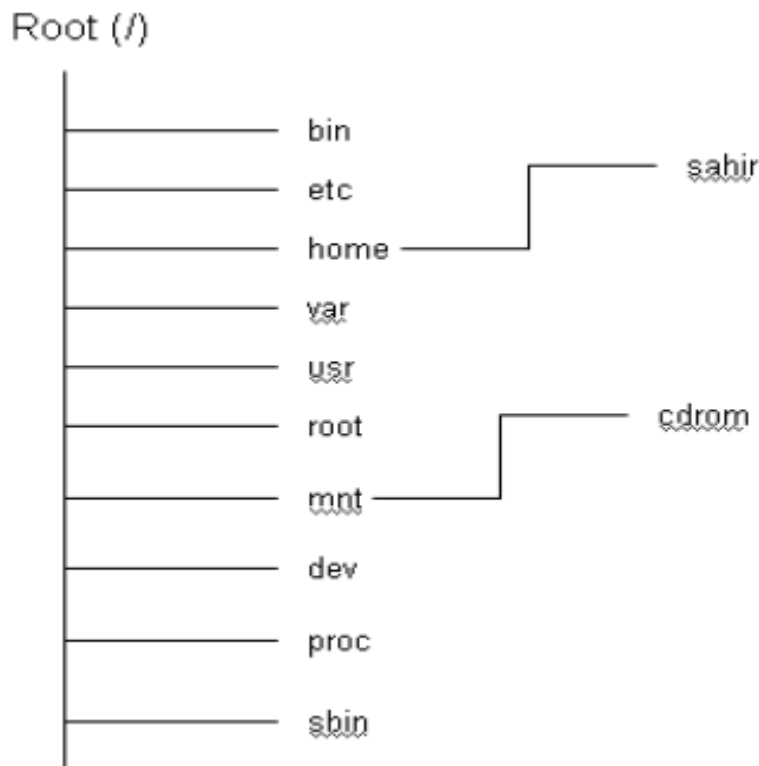


Figure 3.1: File structure in Linux

Figure 3.1 shows the standard folders which are associated with some predefined functionality. The purpose of these folders are summarized in table 3.2.

Folder	Purpose
/bin	This directory contains several useful commands that are used by both the system administrator as well as other users. This directory usually contains the shells like bash, csh as well as much used commands like cp, mv, rm, cat, ls.
/etc	This directory contains all the configuration files for the system.
/home	This is the home directory of a user, which can be found under /home/username.
/var	This directory stores temporary and variable files created by users, such as log files, mail queues, temporary file downloaded from the internet.
/usr	This directory contains all the user binaries. The X system and its supporting libraries can be found here. User programs like telnet, ftp are also placed here.
/root	This is the home directory of the system administrator (called super-user or root).
/mnt	This directory usually contains mount points or sub-directories where you mount your floppy and your CD.
/dev	This is used to store the device files.
/proc	This directory which is not physically present on disk, is a real time, memory resident file system that tracks the processes running on your machine and the status of the system resources.
/sbin	The functionality of this directory is similar to the /bin directory but with the difference that it is only accessible to the 'root' user.

Table 3.2 Folders in the file system in Linux

The root is actually a directory, and it has a number of sub-directories under it. These sub-directories, in turn, have more sub-directories and other files under it. Every file, apart from the root, must have a parent directory, and it should

be possible to trace the ultimate parentage of a file to root. Thus, in Figure 3.1, root is called the **parent directory** of the directories such as bin, usr, var, and so on. When a user logs in, the user is placed in a specific directory. This directory is known as the user's **current directory**.

### ➤ Mounting and unmounting file systems

Mounting refers to a process of making the file system available for use. Mounting the file system makes the files appear as if they are located on the local machine. The `mount` command makes a file system available for use.

#### Syntax:

```
[student@localhost ~]$ mount [-f] [ -n Node] [-o Options] [-p] [-r] [-v VfsName ] [
-t Type | [Device | Node:Directory ] Directory | all | -a ] [ -V [generic_options]
special_mount_points ]
```

The `mount` command instructs the operating system to make a file system available for use at a specified location, called the mount point. After the `mount` command has completed the execution, the specified directory becomes the root directory of the newly mounted file system.

The name of the block device to be mounted is specified as the `Device` parameter. The directory on which the file system is to be mounted, when mounting a device, is specified as the `Directory` parameter.

The `mount` command recognizes five values for the `mount` attributes. They are: `automatic`, `true`, `false`, `removable`, and `readonly`. The arguments of the `mount` command are described in table 3.3.

Option	Function
-f	Denotes the file system type on which to operate
-n	Specifies the remote node that holds the directory to be mounted
-p	Prints the list of mounted file systems in the <code>/etc/vfstab</code> format
-r	Mounts the file system as read-only
-v VfsName	Specifies that the file system is defined by the <code>VfsName</code> parameter in the <code>/etc/vfs</code> file

**Table 3.3: Arguments of the mount command**

The code in Code Snippet 1 mounts a CD-ROM in the directory `/mnt/cdrom`.

#### Code Snippet 1:

```
[student@localhost ~]$ mount /dev/cdrom /mnt/cdrom
```

The `umount` command is used to remove the previously mounted file.

#### Syntax:

```
[student@localhost ~]$ umount [Options] Mount point or Device name of the file
system
```

The common options available with the `umount` command are summarized in table 3.4.

Option	Function
-f	Forces an unmount in a remote environment
-a	Unmounts all mounted file systems
all	Unmounts all mounted file systems
allr	Unmounts all remotely mounted file systems

**Table 3.4: Options of the umount command**

The code in Code Snippet 2 unmounts all the files in a CD-ROM.

**Code Snippet 2:**

```
[student@localhost ~]$ umount /dev/cdrom
```

## 3.2 Manipulating and Searching Commands

In order to organize the information and manage how the system works and looks, the user can create, delete and move files and directories on the hard disk. There are various commands available to create, delete, move, and copy files and directories in Linux.

### 3.2.1 File and Directory Creation and Removal Commands

The working directory or current working directory is the directory you are currently working in. When you login to Linux, your home directory is your working directory. Many Linux commands are concerned with manipulating files and directories. Some of the common commands in their most common forms are discussed in this session.

#### ➤ Creating a directory with the `mkdir` command

This command is used to create a new directory. A directory or file name cannot be more than 255 characters and can contain any characters except forward-slash, `/`. But it is unwise to use certain special characters in filenames such as `$`, `<`, `>` symbols and so on. Filenames and directory names are case sensitive.

**Syntax:**

```
[student@localhost ~]$ mkdir Directory_name
```

The user can use the `mkdir` command to create a new directory under the existing directory. The code in Code Snippet 3 creates a directory `prog-files` under the `temp` directory.

**Code Snippet 3:**

```
[student@localhost ~]$ mkdir temp/prog-files
```

#### ➤ Changing directories with the `cd` command

The user can move around in the file system by using the `cd` (change directory) command. It changes the current directory to the directory specified as the argument. You can use an absolute or relative pathname as an argument.

Absolute pathname begins with forward-slash and contains the names of all the directories that must be traversed to reach the named object.

The relative pathname does not begin with forward-slash and it specifies the name of each directory that must be traversed from the current directory to reach the named object.

**Syntax:**

```
[student@localhost ~]$ cd [Directory_name]
```

The code in Code Snippet 4, uses a relative pathname to traverse to the `progs` directory if it exists.

**Code Snippet 4:**

```
[student@localhost ~]$ cd progs
```

The `cd ..` command is used to move to the parent directory of the current directory. The `cd` command without any argument is used to move to the home directory of the current user. The code in Code Snippet 5 displays how to move one directory up into the `home/users/james` directory.

**Code Snippet 5:**

```
[student@localhost ~]$ cd ../home/users/james
```

➤ **Deleting a directory with the rmdir command**

This command is used to remove a directory.

**Syntax:**

```
[student@localhost ~]$ rmdir [Directory_name]
```

The directory to be removed must be empty before invoking the `rmdir` command. If the directory is not empty, an error message is displayed. The code in Code Snippet 6 removes a directory `prog-files` that is within the `temp` directory.

**Code Snippet 6:**

```
[student@localhost ~]$ rmdir /temp/prog-files
```

➤ **Deleting a file with the rm command**

This command is used to delete files and directories from the file system.

**Syntax:**

```
[student@localhost ~]$ rm [Options] {File(s)}
```

The common options available with the `rm` command are summarized in table 3.5.

Option	Function
-f	Ignores missing files and overrides any confirmation prompts ("force")
-r	Deletes the directory trees recursively
-i	Confirms with the user before deleting the file

**Table 3.5: Options of the rm command**

The code in Code Snippet 7 deletes the file named `linuxbasics.txt`.

**Code Snippet 7:**

```
[student@localhost ~]$ rm -r linuxbasics.txt
```

### 3.2.2 File Manipulation Commands

Linux provides commands for manipulating files as well. Some of these are discussed in detail below.

➤ **Renaming or moving a file with the mv command**

This command can be used for two purposes: first, to move a file from one directory to another, and second, to rename a file or directory.

**Syntax:**

```
[student@localhost ~]$ mv [options] source destination
```

The common options available with the `mv` command are summarized in table 3.6.

Option	Function
-i	Confirms with the user whether the file can be overwritten before renaming
-b	Creates a backup of the file or directory that is to be renamed

**Table 3.6: Options of the mv command**

The code in Code Snippet 8 moves the file `linuxbasics.txt` to the directory `progs` and asks the user whether the file is



to be overridden if it already exists in the directory.

#### Code Snippet 8:

```
[student@localhost ~]$ mv -i linuxbasics.txt progs
[student@localhost ~]$ tree
.
|--Desktop
|--progs
|   |-- linuxbasics.txt
|--temp
```

#### ➤ Copying a file or directory with cp command

This command is used to copy the contents of a source file or directory to a target file or directory.

##### Syntax:

```
[student@localhost ~]$ cp [options] <source file> <target file>
```

The common options available with the `cp` command are summarized in table 3.7.

Option	Function
-i	Prompts the user before overwriting
-r	Copies the entire subdirectory
-b	Creates a backup copy of each destination file if the target file is to be overwritten
-u	Copies only when the source file has been updated more recently than the destination file, or when the destination file is missing

**Table 3.7: Options of the cp command**

The code in Code Snippet 9 copies all the files, directories, and subdirectories from the `progs` directory into the `temp` directory.

#### Code Snippet 9:

```
[student@localhost ~]$ cp -r progs/* temp
[student@localhost ~]$ tree
.
|--Desktop
|--progs
|   |-- linuxbasics.txt
|--temp
|   |-- linuxbasics.txt
```

#### ➤ Comparing two files with the cmp command

This command is used to compare the contents of two files and report the first character that differs.

##### Syntax:

```
[student@localhost ~]$ cmp [options] Filename1 Filename2
```

The common options available with the `cmp` command are summarized in table 3.8.

Option	Function
-l	Displays the byte number (decimal) and the differing bytes (octal) for each difference
-s	Returns only an exit value

**Table 3.8: Options of the cmp command**

The code in Code Snippet 10 compares files `linuxbasics.txt` from `progs` and `temp` directories and displays a detailed list of the byte number and the differing bytes in octal for each character that differs. The `progs/linuxbasics.txt` file contains the data `Hello Student` and `temp/linuxbasics.txt` file contains the data `Hello`.

**Code Snippet 10:**

```
[student@localhost ~]$ cmp -l progs/linuxbasics.txt temp/linuxbasics.txt
7 123 12
cmp: EOF on temp/linuxbasics.txt
```

The output in Code Snippet 10 specifies that the 7th byte is differing in the files. The differing bytes octal numbers are: 123 in `progs/linuxbasics.txt` and 12 in `temp/linuxbasics.txt`.

➤ **Finding what is common among files with the comm command**

This command requires two sorted files. It compares each line of the first file with its corresponding line in the second file.

**Syntax:**

```
[student@localhost ~]$ comm [-1] [-2] [-3] filename1 filename2
```

The options available with the `comm` command are summarized in table 3.9.

Option	Function
-1	Suppresses the output column of lines unique to file1
-2	Suppresses the output column of lines unique to file2
-3	Suppresses the output column of lines duplicated in file1 and file2
--help	Displays the help information and exits
--version	Displays the version information and exits

**Table 3.9: Options of the comm command**

The code in Code Snippet 11 selects lines not common to `linuxbasics.txt` files from `progs` and `temp` directories.

**Code Snippet 11:**

```
[student@localhost ~]$ comm -3 progs/linuxbasics.txt temp/linuxbasics.txt
Hello
Hello Student
```

➤ **Displaying file differences with the diff command**

This command is also used to display the differences between two files. This command can also be used to compare a copy of standard input to itself. If `filename1` is a directory and `filename2` is a file and not a directory, the `diff` command compares the file with the same name as `filename2` in the directory `filename1`, and vice versa. If both `filename1` and `filename2` are directories, the `diff` command compares the corresponding files in both directories, in alphabetical order. The `diff` command never compares the actual contents of a directory as if it were a file.

**Syntax:**

```
[student@localhost ~]$ diff [option] filename1 filename2
```

The options available with the `diff` command are summarized in table 3.10.

Option	Function
-b	Ignores blank lines in files
-i	Ignores the case i.e uppercase and lowercase are considered the same
-s	Reports when two files are same
-w	Ignores spaces and tabs in files when comparing

**Table 3.10: Options of the diff command**

The code in Code Snippet 12 compares two files and ignores the differences in the white space.

**Code Snippet 12:**

```
[student@localhost ~]$ diff -w progs/linuxbasics.txt temp/linuxbasics.txt
lcl
<Hello Student
---
>Hello
```

The output of the `diff` command in Code Snippet 12 implies that the first line is different in each file.

### 3.2.3 Searching for a Pattern Using the grep Command

The `grep` command searches the file for a specified pattern and displays the lines that match the pattern. This command can be used for processing standard input.

**Syntax:**

```
[student@localhost ~]$ grep [option(s)] pattern [file(s)]
```

When no filename has been specified with the `grep` command, it searches the standard text typed in using the keyboard, with the pattern text specified and displays all the matching lines on the standard output. The common options available with the `grep` command are summarized in table 3.11.

Command	Function
-v	Displays all the lines that do not contain the specified pattern
-n	Precedes all the lines with the line numbers
-c	Displays the total number of lines that match the pattern
-r	Performs a recursive search of files starting from the current directory
-i	Ignores the case of the letter when searching
-l	Returns the names of the files that contain atleast one line of the matching pattern
-f	Obtains patterns from file, one per line.

**Table 3.11: Options of the grep command**

The code in Code Snippet 13 displays all the lines that do not contain the word 'bar'.

**Code Snippet 13:**

```
[student@localhost ~]$ grep -v bar progs/linuxbasics.txt
Hello Student
```

The code in Code Snippet 14 counts the number of times the word "Hello" exists in the linuxbasics.txt file.

**Code Snippet 14:**

```
[student@localhost ~]$ grep -c 'Hello' progs/linuxbasics.txt
1
```

The code in Code Snippet 15 searches recursively through the 'progs' directory to find files having the word "Hello".

**Code Snippet 15:**

```
[student@localhost ~]$ grep -r 'Hello' progs
progs/linuxbasics.txt: Hello Student
```

The code in Code Snippet 16 displays all lines in 'temp/linuxbasics.txt' that do not contain any of the words listed in 'progs/linuxbasics.txt', regardless of case.

**Code Snippet 16:**

```
[student@localhost ~]$ grep -vif progs/linuxbasics.txt temp/linuxbasics.txt
Hello
```

### 3.3 Compressing and Uncompressing Files

Sometimes, a group of files need to be stored together in order to share them, transfer them to some other computer, or to take a backup. Grouping of files together is termed as archiving. Sometimes, files need to be packed together into one file so that they occupy less space on the disk. This is termed as compressing, and is quite different from archiving. An archived file, therefore, is not the same as a compressed file; it is just a group of files stored together. It is just a group of files stored together. The size of the files in an archive do not change on the hard disk.

#### 3.3.1 Creating archive with tar command

This command is used to create an archive file that contains directories and files. Archives created with the `tar` command are denoted as tarballs.

**Syntax:**

```
[student@localhost ~]$ tar [option(s)] archive_name file_name
```

The `archive_name` is the name of the archive to be created and `file_names` are the names of the files to be added to the archive. The new archive file created will have an extension of `.tar`.

The common options available with the `tar` command are summarized in table 3.12.

Option	Function
-c	Creates an archive file
-f	Specifies that the next argument in the command will be the name of the archive file to be created
-t	Lists the files in the order in which they should be present in the archive file
-z	Compresses the archived file using gzip compression

**Table 3.12: Options of the tar command**

**Code Snippet 17:**

```
[student@localhost ~]$ tar -cf file.tar file1 file2 file3
```

The code in Code Snippet 17 creates an archive file called `file.tar` from the three files named `file1`, `file2` and `file3` that are located in the current directory.

### 3.3.2 Compressing Files with the gzip Command

This command compresses a single file, appends a .gz extension, and deletes the original file.

#### Syntax:

```
[student@localhost ~]$ gzip [options] filename(s)
```

The common options available with the `gzip` command are summarized in table 3.13.

Option	Function
-d	Uncompresses the file
-t	Tests the integrity of the compressed file
-9	Performs maximum compression

**Table 3.13: Options of the gzip command**

The code in Code Snippet 18 compresses the file, sales to its maximum.

#### Code Snippet 18:

```
[student@localhost ~]$ gzip -9 sales
```

### 3.3.3 Compressing Files with the bzip2 Command

This command compresses and decompresses several files, appends a .bz2 extension, and deletes the original files. File compression and decompression in `bzip2` is similar to `gzip`. However, `bzip2` uses a different algorithm and encoding method that provides better compression logic.

#### Syntax:

```
[student@localhost ~]$ bzip2 [options] filename(s)
```

The files compressed using `bzip2` command are compressed to a much smaller size as compared to the `gzip` command, since it uses high compression ratio. The common options available with the `bzip2` command are summarized in table 3.14.

Option	Function
-d	Uncompresses the file
-c	Compresses or decompresses to standard output
-t	Tests the integrity of the compressed file

**Table 3.14: Options of the bzip2 command**

The code in Code Snippet 19 uncompresses the file sales.

#### Code Snippet 19:

```
[student@localhost ~]$ bzip2 -d sales.doc
```

## 3.4 The vi and vim Editors

Text editors are programs used for editing the text files. The `vi` (visual editor), an interactive text editor, is a commonly used editor in Linux. The `vi` editor allows a user to create, modify, and store files on the computer via a terminal.

The `vim` (vi Improved) is also a text editor created as an extension to the `vi` editor. The `vim` editor is a cross -platform, powerful, small, fast, simple to use, and efficient editor.

### 3.4.1 Basic vi/vim Commands

Linux provides commands to use vi and vim editors. They are discussed below.

#### ➤ Starting the vi editor and editing files

The `vi` command is used to start the vi editor and to edit the named session.

##### Syntax:

```
[student@localhost ~]$ vi filename
```

The `filename` specified as the argument represents the file to be edited. A temporary buffer is allocated by the vi editor as a working space for use during editing.

The `vi` command edits the file specified in the `filename` argument if the file already exists. Otherwise, it creates a new file with the name specified in the `filename` argument.

#### ➤ Modes in vi editor

The vi editor operates in three different modes. They are as follows:

- ✧ **Command Mode** : The vi editor is in command mode when it is started. It is also known as home mode. Some of the actions that can be performed are: moving the cursor, modifying text, deleting text.
- ✧ **Insert Mode** : Insert mode can be activated by typing the letter 'i' without the quotes. In this mode, the typed data is inserted in the file at the cursor position. The data can be appended to the file by typing the letter 'a' (append). The **ESC** key is used to quit from the insert mode and return back to the command mode.
- ✧ **ex Mode** : In this mode, extended commands are given to the vi editor. This mode is also referred to as the **Last Line Mode**. The extended commands include commands for saving, exiting, and searching. To move to the ex mode, the user has to type the character ":" in the command mode.

#### ➤ Commands for moving cursors

In vi editor, the mouse cannot be used to move the cursor within the vi editor screen. The key commands for moving the cursor are listed in table 3.15.

Key	Action
h	Moves the cursor to the left by one character
j	Moves the cursor down by one line
k	Moves the cursor up by one line
l	Moves the cursor to the right by one character
\$	Moves the cursor to the end of current line
^	Moves the cursor to the beginning of current line
G	Moves the cursor to the end of file
Enter	Moves the cursor to the beginning of the next line
:n	Moves the cursor to the line n, where n specifies the line number
w	Moves the cursor to the beginning of the next word
e	Moves the cursor to the end of the next word
b	Moves the cursor to the beginning of the previous word
Ctrl-b	Scrolls the cursor one page up
Ctrl-f	Scrolls the cursor one page down

**Table 3.15: Key commands for moving the cursor**

### ➤ Commands to add, change, and delete the text

Unlike as in other editors, text here cannot be selected and then deleted or overwritten. The vi editor provides commands for manipulating, inserting and deleting the text when it is in insert mode. To switch to command mode from insert mode, the ESC key is pressed. The commands for inserting the text are listed in table 3.16.

Command	Function
i	Inserts the text before the cursor
I	Inserts the text at the beginning of the current line
a	Appends the text after the cursor
A	Appends the text to the end of the current line
o	Opens a new line before the current line and inserts the text in the new line
O	Opens a new line after the current line and inserts the text in the new line

**Table 3.16: Commands to add, change, and delete the text**

The commands for modifying the text are listed in table 3.17.

Command	Function
r	Replaces the current character
R	Replaces text, end with Escape
cw	Changes the current word with new text
c	Changes the characters in the current line
cc	Changes the current line

**Table 3.17: Commands for changing the text**

The commands for deleting the text are listed in table 3.18.

Command	Function
x	Deletes the current character
Nx	Deletes N characters, starting from the current cursor position
dw	Deletes the current single word
D	Deletes the remaining line, starting from the current cursor position
dd	Deletes the current line

**Table 3.18: Commands for deleting the text**

The commands for cutting and pasting the text are listed in table 3.19.

Command	Function
yy	Copies (Yanks) the current line into the buffer
y\$	Copies (Yanks) to the end of the current line from the cursor
yw	Copies (Yanks) from the cursor to the end of the current word
p	Pastes below the cursor
P	Pastes above the cursor
u	Performs undo on the last change
U	Restores the line
J	Joins next line down to the end of the current line

**Table 3.19: Commands to cut and paste the text**

### ➤ Commands for searching the text

Text editing requires searching and replacing one word or phrase with another.

The commands for searching the text are listed in table 3.20.

Command	Function
/text	Searches downwards for the occurrence of text
?text	Searches upwards for the occurrence of text
n	Continues the search in the same direction for the next match
N	Continues the search in the reverse direction for the next match

**Table 3.20: Commands for searching the text**

### ➤ Commands for saving and exiting the vi editor

The commands for saving the working file and/or exiting the vi editor are listed in table 3.21.

Command	Function
:wq	Writes the file to the disk and quits
:q!	Quits without saving the changes
:w! newfile	Writes all lines from the entire current file into the file overwriting any existing file
:n,m w! newfile	Writes the lines from n to m, inclusive, into the file, thereby overwriting any existing file

**Table 3.21: Commands for saving and exiting vi editor**

## 3.4.2 Advanced vi/vim Commands

Linux provides advanced commands to use vi and vim editors. They are discussed below.

### ➤ Visual mode commands

In visual mode, the user can highlight characters, lines, or even blocks and perform actions on the highlighted data such as cut, copy, paste, and delete much like the mouse highlight operations.

The commands used to start the visual mode are listed in table 3.22.

Key	Action
v	Starts the character oriented highlighting
V	Starts the line oriented highlighting
<Ctrl v>	Starts the block oriented highlighting

**Table 3.22: Commands to start the visual mode**

Once you are in visual mode, the cursor keys can be moved to highlight a section of text. After the text has been highlighted, the action key should be typed for the action to take place on the highlighted text.

The different action keys with their actions are listed in table 3.23.

Key	Action
c	Changes the selected text
d	Deletes the selected text
y	Copies the selected text



Key	Action
>	Indents the selected text
<	Removes the indent of the selected text

Table 3.23: Action keys

### ➤ Reading and saving file commands

The user can read the other files in the current file using the `:r` command. That is, the `:r` command will read the external file and place it in the current cursor position.

The output of the command can be read using the `!commandname` after the `:r` command. For example, if the user wants to read the output of the `date` command, then the command to do so is:

```
:r !date
```

Basic commands can be used to save the complete file. But the user can also save some part of the file into some other file. For example, suppose a user wants to save line 16 to line 20 to a file named `newfile`. To perform this, the user should execute the command:

```
:16, 20w newfile
```

### ➤ Configuring vi and vim commands

There are various commands available to configure vi and vim editors. Some of the common ones are listed in table 3.24.

Option	Function
<code>:set</code>	Views limited configuration items
<code>:set all</code>	Views all the configuration items
<code>:set showmatch</code>	Specifies that when right parenthesis is typed, the cursor jumps to the matching left parenthesis
<code>:set autoindent</code>	Specifies that when one line is indented, the new lines below are equally indented
<code>:set textwidth=80</code>	Specifies that when the text exceeds 80 characters, the text is wrapped
<code>:set number</code>	Displays line numbers on the left margin
<code>:set ignorecase</code>	Searches the text regardless of the case

Table 3.24: Commands to configure the vi and vim editors

## 3.5 String Processing

Processing strings involves counting the number of character in a file, displaying specified lines from a file, sorting the text data, removing duplication of text data, removing specified data from the file, combining files, and spell checking.

### 3.5.1 String Processing Commands

There are various string processing commands available in Linux. Some of them are discussed below.

#### ➤ Displaying specified lines from a file using head and tail commands

To display the first few lines of the file, the `head` command is used. The default number of lines displayed is 10.

**Syntax:**

```
[student@localhost ~]$ head <filename>
```

To manually specify the number of lines to be displayed at a time, the argument `n` is used with the `head` command.

**Syntax:**

```
[student@localhost ~]$ head -n <filename>
```

To display the last few lines of the file, the `tail` command is used. The default number of lines displayed is 10.

**Syntax:**

```
[student@localhost ~]$ tail <filename>
```

The `tail` command is mostly used by the system administrators to view the recent entries of log files.

The common options available with the `tail` command are summarized in table 3.25.

Option	Function
-n	Specifies the number of lines to be displayed
-f	Displays the additions to the end of the file

**Table 3.25: Options of the tail command**

To display the additions to the end of the file in real time (that is, as they occur), the `-f` argument is used with the `tail` command. This feature is used by the system administrator for checking the log file.

The code in Code Snippet 20 lists the last 100 lines in the file `linuxbasics.txt`.

**Code Snippet 20:**

```
[student@localhost ~]$ tail -n 100 linuxbasics.txt
```

➤ **Counting the number of characters, words lines using the wc command**

This command is used to count the number of lines, bytes, words, and characters in a file.

**Syntax:**

```
[student@localhost ~]$ wc [options] <filename(s)>
```

After the execution of `wc` command the name of the file is displayed along with the count. The common options available with the `wc` command are summarized in table 3.26.

Option	Function
-c	Counts the number of bytes. If this option is used with <code>-k</code> option, it counts the number of characters
-m	Counts the number of characters
-l	Counts the number of lines
-w	Counts the number of words

**Table 3.26: Options of the wc command**

If more than one filename is specified with the `wc` command, it reports the total count of all the files.

**Code Snippet 21:**

```
[student@localhost ~]$ wc progs/linuxbasics.txt
1      2     15  progs/linuxbasics.txt
```

The output of the command indicates that there is 1 line, 2 words and 15 characters in the linuxbasics.txt file.

### ➤ Sorting the text with sort command

This command can be used to either sort the contents of the file as per a specified format, or merge already sorted files or check whether the files have been sorted or not.

#### Syntax:

```
[student@localhost ~]$ sort [options] <filename(s)>
```

If more than one file is specified in the `sort` command, it concatenates the files and sorts them as one file. The common options available with the `sort` command are summarized in table 3.27.

Option	Function
-n	Sorts according to the numeric value of the string or letter
-f	Changes all lowercase letters to uppercase before comparison
-u	Removes the duplicate lines in the output
-t Character	Uses the character specified in Character element as a field separator
-k POS1	Sorts from the POS1 till the end of file
-k POS1, POS2	Sorts from POS1 to POS2
-r	Reverses the order of sorting
-d	Sorts using dictionary order

**Table 3.27: Options of the sort command**

The code in Code Snippet 22 sorts the file, linuxbasics.txt in a reverse order. The progs/linuxbasics.txt file has the data

Welcome

Hello Student

#### Code Snippet 22:

```
[student@localhost ~]$ sort -r progs/linuxbasics.txt
Welcome
Hello Student
```

The code in Code Snippet 23 changes all the lowercase letters to uppercase and then sorts using the dictionary order.

#### Code Snippet 23:

```
[student@localhost ~]$ sort -d -f progs/linuxbasics.txt
Hello Student
Welcome
```

### ➤ Deleting duplicate lines in a file with uniq command

To delete duplicate lines from a file, the `uniq` command is used. Before the command can be used on a file, the file content should be sorted. This command compares the adjacent lines. If the adjacent lines are duplicate, the first line is kept and the duplicate lines are removed. The `uniq` command can be applied on fields and columns in a file. If it is applied on a particular field or column, then the file should be sorted on that particular field or column.

#### Syntax:

```
[student@localhost ~]$ uniq [options] <Inputfile>
```

Input file refers to the path name of the input file. The common options available with the `uniq` command are summarized in table 3.28.

Option	Function
-c	Precedes each output line with a count of the number of times the line occurred in the input
-d	Displays only one copy of the lines that are repeated
-u	Displays only the unrepeated lines
-f fields	Ignores the first few fields on each input line when comparing
-s N	Ignores the first N characters or field in a line when comparing

**Table 3.28: Options of the `uniq` command**

The code in Code Snippet 24 removes the duplicate lines in the file `linuxbasics.txt` and stores the results in the file `updatedlinuxbasics.txt`. Suppose the `progs/linuxbasics.txt` file has the following data:

Welcome

Welcome

Hello Student

**Code Snippet 24:**

```
[student@localhost ~]$ uniq progs/linuxbasics.txt > updatedlinuxbasics.txt
[student@localhost ~]$ cat updatedlinuxbasics.txt
Welcome
Hello Student
```

The code in Code Snippet 25 displays only the repeated lines and prints the count of the repeated lines.

**Code Snippet 25:**

```
[student@localhost ~]$ uniq -dc progs/linuxbasics.txt > count.txt
[student@localhost ~]$ cat count.txt
2 Welcome
```

➤ **Cut the field or columns of text and display using the `cut` command**

This command is used to cut bytes, characters, or fields from each line of a file and display it on the standard output.

**Syntax:**

```
[student@localhost ~]$ cut [options] <filename>
```

The common options available with the `cut` command are summarized in table 3.29.

Option	Function
-f list	Specifies a field or a column
-n	Specifies that the characters are not split
-d delim	Specifies field or column delimiter. Default is TAB
-s	Suppresses the lines that do not contain the delimiting character
-c list	Specifies character positions
-b list	Specifies byte positions

**Table 3.29: Options of the `cut` command**

The code in Code Snippet 26 displays columns 1 to 2 of each line and ignores the rest of the line.

**Code Snippet 26:**

```
[student@localhost ~]$ cut -c 1-2 progs/linuxbasics.txt
We
We
He
```

➤ **Merging lines of files together using the paste command**

This command reads a line from each files and merges them together separated by a tab.

**Syntax:**

```
[student@localhost ~]$ paste [options] <filename>
```

The common options available with the `paste` command are summarized in table 3.30.

Option	Function
-s	Combines subsequent lines of the same input file
-d list	Changes the output delimiter

**Table 3.30: Options of the paste command**

The code in Code Snippet 27 takes the input from the file `linuxbasics.txt` from `progs` and `temp` directories respectively and pastes them together to a third file, `merger.txt`.

**Code Snippet 27:**

```
[student@localhost ~]$ paste -d: progs/linuxbasics.txt temp/linuxbasics.txt >
merger.txt
[student@localhost ~]$ cat merger.txt
Welcome:Welcome
Welcome:Hello Student
Hello Student:
:
```

The `:` is used as a delimiter.

➤ **Pipes**

The `who` command produces a list of users, one user per line. If this output is saved in a file and then `wc -l` command is applied to count the number of lines in this file, then this would give the count of the number of users. Linux provides a concept of pipes in which the output of one command is given as an input to another. Pipes create a chain of commands to achieve the desired results. The above sequence of `who` and `wc -l` can be combined into a single instruction using the pipe as in Code Snippet 28.

In the output of Code Snippet 28, the `who` command displays one user currently logged into the system. The `wc` command with `-l` option counts the number of lines in the output of `who` command which is 1. The output of `who` command has been passed as an input to `wc` command.

**Code Snippet 28:**

```
[student@localhost ~]$ who | wc -l
1
```

The code in Code Snippet 29 pipes the output to the `ls` command as an input to the printer.

**Code Snippet 29:**

```
[student@localhost ~]$ ls -l | lpr -P printername
```

### 3.5.2 Spell Checking with aspell

The `aspell` in Linux is an interactive spell checker. It provides a menu driven interface that suggests corrections in spellings. It is a utility to manage dictionaries.

#### Syntax:

```
[student@localhost ~]$ aspell [options] <command>
```

The common options available with the `aspell` command are summarized in table 3.31.

Option	Function
<code>-d,--master=string</code>	Specifies the name of the dictionary to use
<code>-l,--lang=string</code>	Specifies the language to use
<code>-p,--personal=file</code>	Specifies the personal word list file name
<code>--extra-dicts=list</code>	Specifies the extra dictionaries to use

**Table 3.31: Options of the aspell command**

The common commands available with the `aspell` command are summarized in table 3.32.

Command	Function
<code>-?,help</code>	Displays the help message
<code>-c,check file</code>	Spell-checks a file
<code>-v,version</code>	Specifies the version number

**Table 3.32: Commands with the spell command**

The code in Code Snippet 30 runs `aspell` on a file called “linuxbasics.txt”. It sets the language as Swiss German while performing spell check.

#### Code Snippet 30:

```
[student@localhost ~]$ aspell -c -lang=de_CH linuxbasics.txt
```

## The Session In Brief

- The command to attach a file or directory is mount.
- The directory commands are used to manipulate a directory. The mkdir command creates a directory. The rmdir command deletes a directory.
- The grep command searches the file for a specified pattern and displays the lines that matches the pattern.
- The vi is an interactive text editor used for editing the text files. The vim is also a text editor and is an extension to the vi editor.
- The head and tail commands are used to display the first few lines and the last few lines of text in a file respectively.
- The wc command is used to count the number of lines, bytes, words, and characters in a file.
- The sort command is used to sort the file in a specified format, merge files that are already sorted, and check files to determine if they have been sorted.
- The aspell utility in Linux is an interactive spell checker.

## Check Your Progress

1. You can use the \_\_\_\_\_ command to display the contents of a directory page-wise, with all the file attributes.
  - a. ls command
  - b. ls -l
  - c. ls -l or more
  - d. more
2. The \_\_\_\_\_ command is used to count the number of lines, bytes, words, and characters in a file.
  - a. mkdir
  - b. count
  - c. wc
  - d. sort
3. The \_\_\_\_\_ utility provides a menu driven interface that suggests corrections in spellings.
  - a. spell
  - b. aspell
  - c. dictionary
  - d. check
4. The \_\_\_\_\_ key is used to move the cursor to the beginning of the current line in the vi editor.
  - a. ^
  - b. B
  - c. G
  - d. ~
5. The \_\_\_\_\_ command is used to delete duplicate lines in a file.
  - a. sort
  - b. uniq
  - c. cut
  - d. delete



## Working with the File System (Lab)

### Session Objectives:

*At the end of this session, you will be able to -*

- Manipulate files and directories
- Compress and Uncompress files
- Use basic string processing commands

### Part I - 60 Minutes

---

#### Exercise 1: File creation and removal commands

##### Problem

Give the commands to mount MS-DOS floppy, create and remove directories.

##### Analysis

The problem requires the user to mount the MS-DOS floppy, create directories, and remove a directory.

##### Solution

The `mount` command attaches a file system in a device or in the network to a directory on the local system. If the directory has files, those files are hidden from view until the mount is removed. The file `/etc/fstab` contains a list of all the mounts in the local machine. If a line of the `fstab` file contains the `user` option, then a non root user can use the `mount` command to mount it. The user can then specify the name of the device or its mount to access the file system.

1. Login to root user. Type the following command to mount the MS-DOS floppy.

```
[root@localhost ~]$ mount /dev/fd0 /mnt/floppy
```

In this, `/dev/fd0` refers to the floppy drive and `/mnt/floppy` indicates the mount point.

2. Type the following command to umount the MS-DOS floppy.

```
[root@localhost ~]$ umount /dev/fd0  
or  
[root@localhost ~]$ umount /mnt/floppy
```

3. Login to student user. Type the following command to create a file named `student1.txt`.
4. Type the text "Hello Student".
5. Press `Ctrl + Z` to save the text.

```
[student@localhost ~]$ cat > student1.txt
Hello Student

[1]+  Stopped                  cat > student1.txt
```

- 6. Type the following command to create a file named student2.txt.**
- 7. Type the text “ Welcome to Linux Tutorials”.**
- 8. Press Ctrl + Z to save the text.**

```
[student@localhost ~]$ cat > student2.txt
Welcome to Linux Tutorials

[2]+  Stopped                  cat > student2.txt
```

- 9. Type the following command to display the contents of a file called student1.txt.**

```
[student@localhost ~]$ cat student1.txt
Hello Student
[student@localhost ~]$ -
```

- 10. Type the following command to display the contents of more than one file.**

```
[student@localhost ~]$ cat student1.txt student2.txt
Hello Student
Welcome to Linux Tutorials
[student@localhost ~]$ -
```

- 11. Type the following command to create a file in vi editor.**

```
[student@localhost ~]$ vi linuxtest.txt
-
-
-
-
"linuxtest.txt" [New File]
```

- 12. Press i to insert text at the current cursor position.**

Add the following text to the file.

```
hello world!
hello world!
cartoons are good
especially toons like tom (cat)
what
the number one song
12221
they love us
I too
```

**13. Press the ESC key, type :wq to save all the changes and quit.**

**14. Type the following command to delete the file student2.txt**

```
[student@localhost ~]$ rm student2.txt
[student@localhost ~]$
```

**15. Type the following command to create a directory.**

```
[student@localhost ~]$ mkdir prog-files
[student@localhost ~]$ -
```

**16. Type the following command to create a new directory under prog-files directory.**

```
[student@localhost ~]$ mkdir prog-files/linuxsamples
[student@localhost ~]$ -
```

**17. Type the following command to remove the directory linuxsamples.**

```
[student@localhost ~]$ rmdir prog-files/linuxsamples
[student@localhost ~]$ -
```

## Exercise 2: File manipulation commands

### Problem

Give the commands to copy, and move files and directories.

### Analysis

The problem requires `cp` and `mv` commands to copy and rename files and directories.

### Solution

**1. Type the following command to copy the contents of the file student1.txt to the file student2.txt.**

```
[student@localhost ~]$ cp student1.txt student2.txt
[student@localhost ~]$ -
```

The contents of the file student1.txt are copied to file student2.txt. If student2.txt already existed, the data in the file would

be overwritten by the data from student1.txt.

2. Type the mv command to move the file student2.txt to the directory prog-files of the user.
3. Type the tree command to view the directory structure.

```
[student@localhost ~]$ mv student2.txt prog-files
[student@localhost ~]$ tree
.
|-- Desktop
|-- linuxtest.txt
|-- prog-files
    |-- student2.txt
|-- student1.txt
```

4. Type the mv command to move the prog-files directory into a directory called linuxsample directory.
5. Type the tree command to view the directory structure.

```
[student@localhost ~]$ mv prog-files linuxsample
[student@localhost ~]$ tree
.
|-- Desktop
|-- linuxsample
    |-- student2.txt
|-- linuxtest.txt
|-- student1.txt
```

This is effectively the same as renaming the 'prog-files' directory to 'linuxsample'.

### **Exercise 3: Searching for text within the files**

#### **Problem**

Give the commands to search for a text string inside a file.

#### **Analysis**

The problem requires that the user should search the file for a specified pattern and display the lines that match the pattern.

#### **Solution**

1. Type the following command to view the linuxtest.txt file.

```
[student@localhost ~]$ cat linuxtest.txt
hello world!
hello world!
cartoons are good
especially toons like tom (cat)
what
the number one song
12221
they love us
I too
```

**2. Type the following command to look through all the lines for the 'too' pattern.**

```
[student@localhost ~]$ grep "too" linuxtest.txt
cartoons are good
especially toon like tom (cat)
I too
```

The `grep` command prints "too", as well as "cartoons" and "toon" because `grep` treats "too" as an expression. The line is printed if the expression is found.

## **Exercise 4: Compressing and Uncompressing Files**

### **Problem**

Give the commands to compress and uncompress files in Linux.

### **Analysis**

The problem requires the user to use the `tar` command with various options to compress and uncompress the files.

### **Solution**

**1. Type the following command to archive the contents of the linuxsample directory in an archive labeled newtar.tar.**

```
[student@localhost ~]$ tar -cf newtar.tar linuxsample
```

The `-c` option in the `tar` command creates a new archive file. The `-f` option in the `tar` command specifies that the next argument in the `tar` command is the name of the archive file to be created.

**2. Type the following command to remove the linuxsample directory and its contents.**

```
[student@localhost ~]$ rm -r linuxsample
```

**3. Type the following command to extract the contents of the newtar.tar archive.**

```
[student@localhost ~]$ tar -xvf newtar.tar
```

The `-x` option in the `tar` command extracts an archive file. The `-v` option displays the name of each file.

## Exercise 5: String Processing

### Problem

Give the commands to perform the following activities:

- ✧ Count the number of lines in a file
- ✧ Sort the file
- ✧ Merge two files
- ✧ Spell check a file

### Analysis

The problem requires the user to use basic string processing commands such as `wc`, `sort`, `uniq`, `cut`, `paste`, and `aspell` in Linux.

### Solution

1. Type the following command to count the number of lines in the `linuxtest.txt` file.

```
[student@localhost ~]$ wc -l linuxtest.txt
9 linuxtest.txt
```

2. Type the `sort` command to sort the file and place the sorted data in a new file `sorted.txt`.

3. Type the `cat` command to view the `sorted.txt` file.

```
[student@localhost ~]$ sort linuxtest.txt > sorted.txt
[student@localhost ~]$ cat sorted.txt
12221
cartoons are good
especially toon like tom (cat)
hello world!
hello world!
I too
the number one song
they love us
what
```

4. Type the `uniq` command to provide a count of how many times each value appears.

5. Type the `cat` command to view the `uniqdemo.txt` file.

```
[student@localhost ~]$ uniq -c sorted.txt > uniqdemo.txt
[student@localhost ~]$ cat uniqdemo.txt
1 12221
1 cartoons are good
1 especially toon like tom (cat)
2 hello world!
1 I too
1 the number one song
1 they love us
1 what
```

The line containing 'hello world!' is prefixed by the numerical 2. This indicates that the line 'hello world' has been repeated twice in the file uniqdemo.txt.

#### 6. Create two files name.txt and marks.txt with the following data.

name.txt

```
Name
James
Martin
Alex
Lisa
Mark
```

marks.txt

```
Marks
67
55
96
36
67
```

#### 7. Type the following command to display the first 3 columns of the name of student.

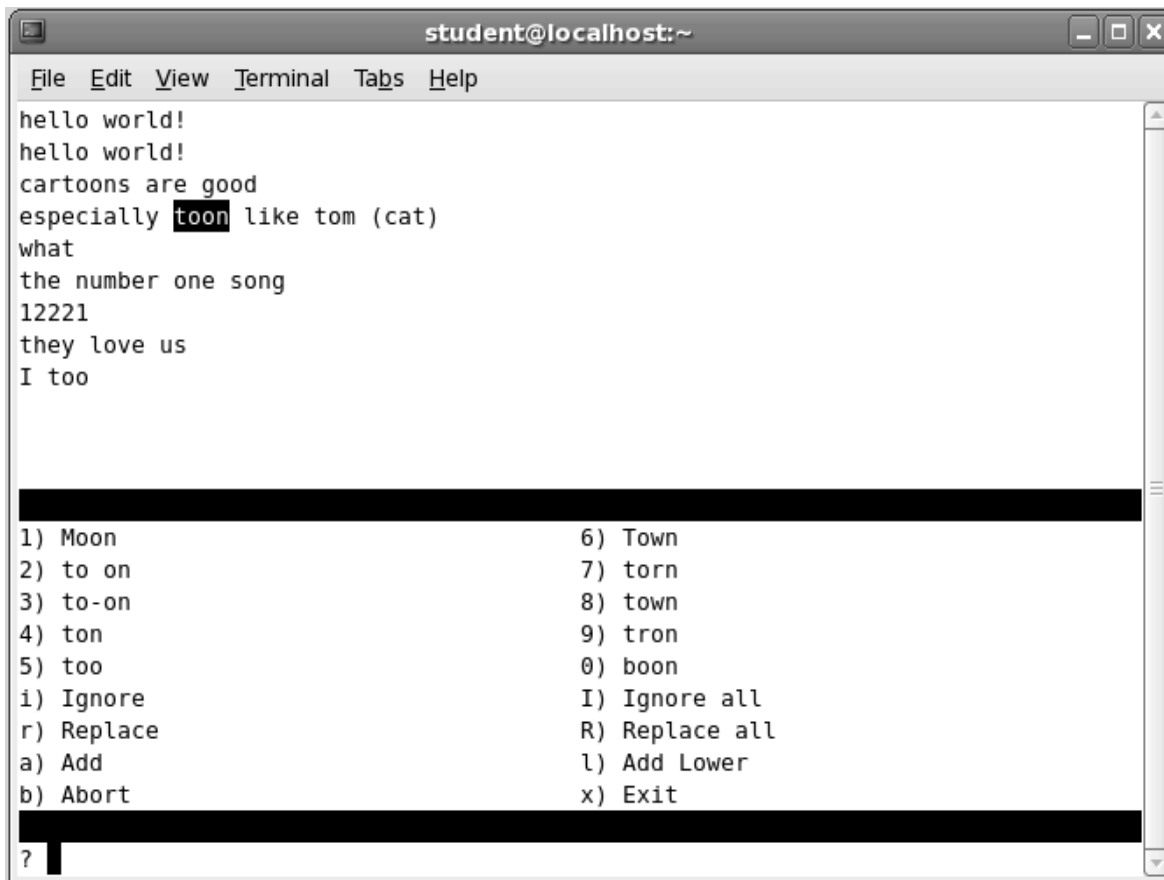
```
[student@localhost ~]$ cut -c 1-3 name.txt
Nam
Jam
Mar
Ale
Lis
Mar
```

#### 8. Use the paste command to merge the files name.txt and marks.txt.

```
[student@localhost ~]$ paste name.txt marks.txt
Name      Marks
James     67
Martin    55
Alex      96
Lisa      36
Mark      67
```

**9. Type the following command to spell check the file linuxtest.txt.**

```
[student@localhost ~]$ aspell -c linuxtest.txt
```



**Figure 4.1: Output of aspell**

The word 'toon' is highlighted in Figure 4.1, and a list of suggested options is provided to the user to choose from. The user can press a number from the given list to select the appropriate spelling.



## Part II - 60 Minutes

---

1. Give the commands to do the following:
  - ✧ Create a directory called LinuxExam
  - ✧ Create a file named subject1.txt under the LinuxExam directory and add some text to the file.
  - ✧ Display the content of the subject1.txt file
  - ✧ Display the content of the directory LinuxExam
  - ✧ Copy the file subject1.txt to subject2.txt
  - ✧ Rename the subject1.txt file to subject3.txt
  - ✧ Rename the directory LinuxExam to Linux
  - ✧ Delete the file subject3.txt
  - ✧ Delete the directory Linux
2. Create a new document linuxbasics.txt using vi editor. Add text to the document. Save this file and quit the editor. Edit the document again, and perform a spell check on the document using appropriate Linux commands.

## Do It Yourself

1. Give the commands to:
  - ✧ Display the contents of a directory page-wise.
  - ✧ Display the contents of all the files whose names end with the character 'd'.
  - ✧ Display the contents of the files 'having file names with only two characters. These files should have names beginning with 'q' and the next letter being 'a', 'b', 'c' - for example, qa, qb, qc.
2. Give the commands to:
  - ✧ Take the backup of the program-files directory in zipped form and store it in another directory. The name of the backup file should be specified as Data.zip.
  - ✧ Compare the data in the existing Data.zip file and the program-files directory; append only those files that are not present in the archive.
  - ✧ Create a backup of the Presentation directory excluding the file "Notes". Ensure that when adding a file to the archive, the user is prompted for confirmation. The backup has to be taken on a floppy disk.

# The bash Shell and bash Shell Scripting

## Session Objectives:

*At the end of this session, you will be able to -*

- Explain what is meant by the bash shell
- Explain how to change the shell
- List the various command line shortcuts
- Explain shell script
- Explain how to generate outputs using shell scripts
- Explain how to use the structured language constructs

## 5.1 The bash Shell

A computer only understands binary language which is composed of two binary numbers, 0 and 1. In early days of computing, the instructions were provided to the computer using binary language. But it is difficult for developers to read and write in binary language. So, the concept of Shell was introduced, in which the operating systems contain a program that accepts the user's instructions or commands in a language very similar to English. If the instructions or commands entered are valid, these are passed on to the kernel. The shell interprets the commands given by the user and translates them into machine code so that the kernel can understand.

### 5.1.1 Introduction to the bash Shell

Shell is an intermediary program that interprets the user-typed commands at the terminal. The shell converts these user typed commands to a form that the kernel understands. Thus, the shell eliminates the need for direct communication between the programmer and the kernel.

In Linux, the shell is the first program that starts when the user logs in, and it keeps running until the user logs out. There are several shells available that can run on the Linux platform, but the default shell of Linux is `bash`. The name `bash` has come from the phrase "Bourne Again Shell".

The shell interprets the commands and executes the requested program. Linux is a multitasking operating system, which means that it is possible to execute more than one program at a time. Linux is also a multi-user OS, which means that it can have more than one shell running at the same time. In a multi-user system, each user gets a copy of the shell after logging in. As a user, you have access only to the programs you are running, not the ones that the other users are running. The programs are kept separate because they are enclosed in a shell.

The `bash` is responsible for command line editing, job control, stream manipulation (piping and redirection), wildcard expansion, aliases, file completion, command history, variables, control structures, sub shells, and so on.

#### ➤ Starting the shell

When the shell starts depends on whether the user uses a graphical or text-mode login. If you are logging on in text-mode, the shell is immediately started after entering the (correct) password. If you are using a graphical login manager such as **gdm**, log on to the Linux system, and the **Terminal** option from the **Accessories** menu.

### 5.1.2 Identifying the shell and changing the shell

When the user logs into a Linux machine, a series of messages, followed by a \$ prompt appears. The ` character preceding the cursor is called the shell prompt; it tells you that the system is ready and waiting for input. As long as the user does not enter something through the keyboard, this prompt remains with the cursor. The `sh` command is located in `/bin` directory. This is the bash Shell.

The shell for any user can be switched temporarily, or it can be changed semi-permanently. Although `bash`, the default shell on Linux, is highly versatile and can be used for almost anything, each shell has its own characteristics and there might be situations in which it is preferable to use some other shell, such as `ash`, `csch`, `ksh`, `sh` or `zsh`. For example, the `csch` shell has a syntax that resembles that of the highly popular C programming language, and thus programmers sometimes prefer it. The current default shell for a user can be determined using the `grep` command. The code in Code Snippet 1 can be used to find the default shell for a user named student.

#### Code Snippet 1:

```
[student@localhost ~]$ grep student /etc/passwd
```

The `grep` returns a line such as `student:513:513:~/home/student:/bin/bash`, which indicates that the shell used by the user student, is `bash`.

It can also be useful to see what shells are available on a particular system. The code in Code Snippet 2 uses the `cat` command to read the `/etc/shells` file, which lists all the shells installed on the system.

#### Code Snippet 2:

```
[student@localhost ~]$ cat /etc/shells
```

To switch to a different shell, type in the shell name at the command prompt and press `Enter`. The code in Code Snippet 3 displays how to change from the current shell to `sh` shell.

#### Code Snippet 3:

```
[student@localhost ~]$ sh
```

Upon changing shells, a different command prompt is shown, depending on the new shell. To return to the original shell, the user needs to type the name of the original shell at the command prompt followed by pressing of the `Enter` key. The code in Code Snippet 4 returns to the `bash` shell from the `sh` shell.

#### Code Snippet 4:

```
[student@localhost ~]# bash
```

The method to permanently change the shell may vary according to the system. However, in general, the first step is to determine the full path, also called the absolute path, to the shell. Once the absolute path to the new shell has been determined, a user can change the default shell using the `chsh` (i.e., change shell) command. The `chsh` command prompts the user to enter the password followed by the absolute path of the desired shell.

### 5.1.3 Command line shortcuts and expansions

The bash shell provides many powerful command line shortcuts and tools that help improve the performance of the bash shell. Some of the command line shortcuts are as follows:

#### ➤ File Globbing

Often, the user is required to use the same commands on more than one file at the same time. The wildcards, or metacharacters allow one pattern to be spread out to multiple filenames. This is termed as **globbing**.

Consider an example in which a directory contains the following files:

```
john.txt  jasmine.txt  birdsing.mp3  song.mp3
```

The codes in Code Snippet 5 and Code Snippet 6 demonstrates how to delete all the files that have the extension .mp3.

#### Code Snippet 5:

```
[student@localhost ~]$ rm *.mp3
```

The output of the Code Snippet 5 is the same as the output of Code Snippet 6. That is, in both the cases, the mp3 files are deleted.

#### Code Snippet 6:

```
[student@localhost ~]$ rm birdsing.mp3 song.mp3
```

The character, ‘ \* ’, used in Code Snippet 5 is termed as a wildcard character. The different wildcard characters available are as follows:

- ✧ \* - matches zero or more characters
- ✧ ? - matches any single character
- ✧ [a - z] - matches a character from a range of characters
- ✧ [^ a - z] - matches all the characters excluding the characters specified in the range

#### ➤ Auto-completing on the command line with the Tab key

Pressing the `Tab` key helps to auto-complete a command name or a file name on the command line. Typing the initial few characters of a command at the command prompt and pressing the `TAB` key completes the command, or the name of an existing directory or file.

For example, typing the command as shown in Code Snippet 7 and pressing the `Tab` key completes the command.

#### Code Snippet 7:

```
[student@localhost ~]$ cd /u
```

The characters “/u” are typed and the `Tab` key is pressed. Then the characters “/s” are added and the `Tab` key is pressed. The output of this is :

```
[student@localhost ~]$ cd /usr/share/
```

Then characters “/f” “o” “n” are typed and the `Tab` key is pressed. Next, typing of “/t” followed by pressing of the `TAB` key and “/d” followed by the pressing of `Tab` key will complete the command. This locates the following path: `/usr/share/fonts/ttf/decoratives.`

Thus, the `TAB` key is used to auto-complete the commands.

#### ➤ Storing history of commands with the bash history

To view the last command typed in by the user, the up arrow key on the keyboard needs to be pressed at the bash prompt. Since the commands are stored in the bash history, one can view all the commands by pressing the up arrow key repeatedly.

Pressing the up-arrow key allows the user to view all the commands typed for that user’s login. If the user wants to view all the stored commands, the `history` command can be used as shown in Code Snippet 8.

#### Code Snippet 8:

```
[student@localhost ~]$ history
```

The bash history also provides a variety of other ways to retrieve the commands from the history list. The shortcuts for retrieving the commands from history are listed in table 5.1.

Command Line Shortcut	Description
!!	Repeats the last command
!c	Repeats the last command that started with the letter 'c'
!n	Repeats a command by its number in the history output
!?abc	Repeats the last command that contains abc
!-n	Repeats a command entered n commands earlier

Table 5.1: History shortcuts

The code in Code Snippet 9 retrieves the eighth command that had been entered by the user.

**Code Snippet 9:**

```
[student@localhost ~]$ !8
```

The code in Code Snippet 10 retrieves the last command that started with the letter V.

**Code Snippet 10:**

```
[student@localhost ~]$ !V
```

Pressing the up-arrow keys and down arrow keys allows the user to scroll through the previously typed commands. To search for stored commands, the <Ctrl+R> key combination can be used.

➤ **Referring to home directory with tilde (~)**

The tilde (~) command is used as a shortcut for referring to the current user's or another user's home directory.

**Syntax:**

```
[student@localhost ~]$ ~username
```

This command will allow the user to switch to switch to the home directory of the user whose user name is specified.

➤ **Explaining the variable and string declaration in bash shell**

The variables in the shell are defined by words on the command line that are prefixed by a \$ sign. The shell substitutes the string with the value of the variable before the command is executed.

The code in Code Snippet 11 first expands the variable \$HOME to its appropriate value, which is the user's home directory, and then executes the cd command. The code in Code Snippet 11 is thus expanded as:

```
[student@localhost ~]$ cd /home/username/index.html
```

**Code Snippet 11:**

```
[student@localhost ~]$ cd $HOME/index.html
```

The set command is used to view the list of variables and their values.

Commas are used to separate the various string patterns. The code in Code Snippet 12 demonstrates the use of commas.

**Code Snippet 12:**

```
[student@localhost ~]$ echo {a,b}
a b
[student@localhost ~]$ echo x{a,b}
xa xb
```

In the first echo statement, there is no value to the left of the curly braces. So the echo statement considers it as

blank and prints “a b”. In the second `echo` statement, the character ‘x’ is placed to the left of the curly braces. The comma between a and b is used to separate strings. The curly braces are therefore expanded to form xa and xb.

### ➤ Command substitution with back quotes

The use of back quotes is termed as command substitution; it allows the output of a command to replace the command itself.

#### Syntax:

```
[student@localhost ~]$ `(command)`
```

The bash expands the command by executing the command, and replaces the command substitution with the result of the command, and deletes the new line. Command substitutions may be nested. To use the back quotes in the command, escape the inner back quotes with backslashes.

The code in Code Snippet 13 executes ‘ls -l’ part first and then substitutes the output after the string “The contents of this directory are “. Then both of these together (directory listing + the string) are written as an output to a file, dir.txt.

#### Code Snippet 13:

```
[student@localhost ~]$ echo "The contents of this directory are "`ls -l`" > dir.txt
```

### ➤ Arithmetic expansion

In arithmetic expansion, the arithmetic expression is evaluated and the result is substituted.

#### Syntax:

```
[student@localhost ~]$(( expression ))
```

The code in Code Snippet 14 uses the backslash before the asterisk to ensure that each element is a separate shell word.

#### Code Snippet 14:

```
[student@localhost ~]$ echo Area : `expr $X \* $Y`
```

Instead of adding the backslash before the shell words, an alternative method would be to use the `$[ ]` syntax to perform mathematical functions.

The code in Code Snippet 14 is rewritten using the `$[]` syntax as shown in Code Snippet 15.

#### Code Snippet 15:

```
[student@localhost ~]$ echo Area: ${ $X * $Y }
```

### ➤ The escape character backslash

A non-quoted backslash ‘\’ is the bash escape character. It preserves the literal value of the next character that follows, except for the newline character. If a \newline pair appears, and the backslash itself is not quoted, the \newline is treated as a line continuation. It means that it is removed from the input stream and effectively ignored.

### ➤ Single and Double quotes

Enclosing characters in single quotes (‘ ’) preserves the literal value of each character within the quotes. A single quote may not occur between single quotes, even when preceded by a backslash.

The characters ‘\$’ and ‘”’ retain their special meaning within double quotes. The backslash retains its special meaning only when followed by one of the following characters: ‘\$’, ‘”’, ‘\’, or newline. Within double quotes, backslashes that are followed by one of these characters are removed. The characters that precede the backslashes are left

## Working with Red Hat Enterprise Linux 5.0

unchanged. To add the character double quotes, it must be preceded with a backslash. If enabled, history expansion will be performed unless an '!' appearing in double quotes is escaped using a backslash. The backslash preceding the '!' is not removed.

The steps in the process of expanding a command line by a bash shell are as follows:

1. The command line is split into the shell words, delimited by spaces, tabs, new lines, and some other characters.
2. Functions are expanded.
3. Curly brace ({} ) statements are expanded.
4. Tilde (~) statements are expanded.
5. Parameters and variables are substituted.
6. The lines are again splitted into shell words.
7. File globs (\*, ?, [abc]) are expanded.
8. The command is executed.

## 5.2 bash Shell Scripting

---

All shell statements in Linux can be entered at the command line. However, when a group of commands need to be executed regularly, it is a good practice to store them in a file. The shell can read the files and execute the commands within the files. Such files are called scripts. In other words, shell script is a text file containing commands. It helps to automate the processes that is executed at the command by making the scripts programmable. For example, suppose everyday after logging in, the user performs the following operations:

- Check the system date
- Check the e-mail
- Look at the calendar for today's schedule

To perform these tasks everyday, the user needs to enter three commands daily. To save the time and automate the task, a shell script containing these commands can be created. Thus, daily, the user needs to enter just one command instead of three commands.

Shell scripts, is a powerful tool for system administration and troubleshooting, which are run mostly during the installation of the operating system.

The three tasks that need to be performed to convert an ordinary file to a script file are as follows:

- It should be executable.
- It should be in text format and contain executable statements.
- It should have the #! comment appearing at the first line indicating the interpreter to be used.

To make a file executable, the `chmod` command is used which changes the access mode of a file. Only the file owner, or the root user, may change the access mode.

### 5.2.1 Creating Shell Scripts

A script file is created by using the vi editor. The code in Code Snippet 16 creates and edits a text file by using the `vi` command.



**Code Snippet 16:**

```
[student@localhost ~]$ vi
```

The user can then press `i` to insert text at the current cursor position. Finally, the user can type the script as shown in Code Snippet 17.

**Code Snippet 17:**

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

After creating the script file, the user should save the file with the name as 'my\_script'. This is done by pressing the `:wq` command followed by a space bar.

Anything following the '#' symbol is considered as a comment and is ignored by the interpreter. Adding comments to the shell script makes the script readable and helps the person to understand what the script does. Thus, it is a good practice to have comments inserted in the script file.

The first line in a shell script should contain 'magic', which is generally referred to as the **shebang**. This line informs the operating system about the interpreter being used to execute the shell script. Some of the shebang examples are as follows:

- `#!/bin/bash` - used for Bash scripts.
- `#!/bin/sh` - used for Bourne shell scripts.
- `#!/bin/csh` - used for C shell scripts.

After creating and saving the shell script, its file permissions need to be changed to make the script executable. The `chmod` command is used to set the permissions on the file. The permission groups are represented as:

- Owner : Represented by 'u'
- Group : Represented by 'g'
- World : Represented by 'o'
- All of the above : Represented by 'a'

**Syntax:**

```
[student@localhost ~]$ chmod u+x scriptfilename
```

The script can be executed by either placing the script file in a directory in the executable path, or by specifying the absolute or relative path to the script file on the command line. The `u` and `x` denote the permissions assigned to the script file. The 'u' specifies the user who owns the file, and 'x' specifies the permission to execute the file.

**5.2.2 Generating outputs with echo and printf commands**

To display some text or the value of a variable, the `echo` command is used.

**Syntax:**

```
[student@localhost ~]$ echo [options] [string, variables...]
```

The common options available with the `echo` command are listed in table 5.2 .

Option	Description
-n	Does not display the trailing new line
-e	Enables interpretation of the backslash escaped characters in the strings

**Table 5.2: Options of the echo command**

The code in Code Snippet 18 demonstrates the use of the `echo` command. The `-n` option does not print the newline after the statement “Please enter the correct login name”.

**Code Snippet 18:**

```
[student@localhost ~]$ echo "Welcome to Red Hat Enterprise Linux"
[student@localhost ~]$ echo -n "Please enter the correct login name"
```

To display the formatted output, the `printf` command is used. The `printf` command does not provide a newline, so multiple `printf` statements can be applied to one line for complex formatting. The newline character, `\n`, is used if a newline is desired.

**Syntax:**

```
[student@localhost ~]$ printf Format [ Argument]
```

The `printf` command converts, formats, and writes its `Argument` parameters to the standard output. The `Format` parameter can consist of literal characters, format control strings and additional options. Format control strings are of the form `%width.precision` followed by a conversion character.

The code in Code Snippet 19 shows an example, in which the result is formatted as `05.2f`, which means the result is a floating point number with a total width of 5 digits.

**Code Snippet 19:**

```
#!/bin/sh
$Result=6.789
printf "The result is %05.2f\n" $Result
```

The output after executing the code in Code Snippet 19 is :

```
The result is 06.79
```

**5.2.3 Reading input with the read command**

The `read` command reads one line of data from the standard input and separates it into individual words. The separated words are assigned to the variables sequentially, that is the first word is assigned to the first variable, the second word is assigned to the second variable and so on. If the number of words exceeds the number of variables, then the remaining words are assigned to the last variable.

**Syntax:**

```
[student@localhost ~]$ read [Options] [filename]
```

The common options available with the `read` command are listed in table 5.3.

Option	Description
-a aname	The words are assigned to sequential indices of the array variable <code>aname</code> , starting at 0
-d delim	The character specified in <code>delim</code> is used to terminate the input line, rather than newline character
-e	If the standard input is coming from a terminal, <code>readline</code> is used to obtain the line
-n nchars	Read returns after reading <code>n</code> number of characters
-s	Silent mode. If input is coming from a terminal, characters are not echoed

**Table 5.3: Options of the read command**

The code in Code Snippet 20 prompts the user to enter three values, and reads the values entered by the user.

**Code Snippet 20:**

```
#!/bin/bash
read "Enter three values:" a b c
echo "Value of a is $a"
echo "Value of b is $b"
echo "Value of c is $c"
```

**5.2.4 bash Shell Variables**

Like every programming language, the shell offers the facility to define and use the variables in the command line. These variables are known as shell variables. Shell variables are assigned a value using the = operator. To access the value of the variable, the \$ character is used as a prefix before the variable name. The code in Code Snippet 21 demonstrates the use of shell variables.

In Code Snippet 21 a variable named 'x' is assigned a value '40'. The value of the variable is displayed to the standard output using the `echo` command and prefixing the variable name with the '\$' symbol.

**Code Snippet 21:**

```
[student@localhost ~]$ x=40
[student@localhost ~]$ echo $x
```

The general form of declaring a shell variable is: `variable=value`. The variables are of string type, that is the value is stored in an ASCII format. Any word preceded by a \$ sign is considered as a variable by the shell. The variable is then replaced by the value assigned to it. All the shell variables are initialized to null strings by default.

To assign multi-word strings to a variable, the user should enclose the value within single quote as shown in Code Snippet 22.

**Code Snippet 22:**

```
[student@localhost ~]$ x='Welcome to Linux'
[student@localhost ~]$ echo $x
```

Variables created within a shell are local to that shell. They are not accessible to the other shells. The `set` command shows a list of all variables currently defined in a shell. If a variable is to be accessible to commands outside the shell, use the `export` command to export it into the environment.

Shell procedures can accept arguments from the command line. When arguments are specified with a shell procedure, they are assigned to certain variables called **positional parameters**. The first argument is read by the shell into the parameter \$1, the second argument into \$2, and so on.

The code in Code Snippet 23 displays the name of the program which is assigned to the argument \$0, and the first argument which is assigned to the argument \$1.

**Code Snippet 23:**

```
[student@localhost ~]$ echo "The program name is $0"
[student@localhost ~]$ echo "The first argument is $1"
```

**5.2.5 The expr command**

This command performs two functions. It can perform arithmetic operations on integers, and also manipulate strings to a limited extent. The `expr` command can perform the four basic arithmetic operations, as well as the modulus (remainder)

## Working with Red Hat Enterprise Linux 5.0

function. However, this command can handle only integers. Some of the examples showing the usage of the `expr` command, along with their outputs are shown in Code Snippet 24.

### Code Snippet 24:

```
[student@localhost ~]$ x=3
[student@localhost ~] y=5                                # Variable assignments;

[student@localhost ~]$ expr 3+5
Output :
8

[student@localhost ~]$ expr 3 \* 5                        # The Asterisk is escaped here
Output:
15

[student@localhost ~]$ expr $x - $y
Output:
-2

[student@localhost ~]$ expr $y / $x                      # The decimal part after the division
                                                         operation is truncated
Output:
1

[student@localhost ~]$ expr 13 % 5
Output:
3
```

## 5.3 Structured Language Constructs

Structured programming is an organized approach to programming that uses a hierarchy of modules. Modularization means grouping of statements together (making modules) that have some relation to each other. In other words, the programs are broken into logical functional section, thereby making it easier to write, debug, understand and maintain the code. Shell programming follows the structured programming model. Therefore, as in structured programming, shell programming also supports the following three types of control structures:

- Sequential - Executes the code line by line till the end of the program
- Selection - Executes the appropriate code based on a logical decision
- Repetition - Repeatedly executes the code based on a logical decision

### 5.3.1 The if / else statements

The `if` statement evaluates the expression to a boolean value of `true` or `false`. If the expression evaluates to `true`, then the commands after the `then` statement are executed. If the expression evaluates to `false`, the commands between the `then` and `fi` statements are skipped and the shell resumes processing the statements resumes with the processing of the statements lying outside the `if` block.

#### Syntax:

```
if [ condition ]; then
    command1
    command2
    ....
fi
```

The code in Code Snippet 25 checks if keyboard input is being accepted. If keyboard input is being accepted then the message is displayed on the standard output.

#### Code Snippet 25:

```
if tty -s; then
    echo "Enter text and end with ^D"
fi
```

The `if - then` statement, evaluates the code following the `then` statement only if the condition is `true`. The limitation of the `if-then` statement is that it does not consider the situation if the expression evaluates to `false`.

The above problem can be solved using the `if / else` statement.

#### Syntax:

```
if condition
then
    do something
else
    do something else
fi
```

The code in Code Snippet 26 searches for a pattern 'director' in the file `emp.lst`. If the pattern is found, it displays the message "Pattern found", otherwise it displays the message, "Pattern not found".

#### Code Snippet 26:

```
#!/bin/sh
if grep "director" emp.lst
then
    echo "Pattern found"
else
    echo "Pattern not found"
fi
```

In Code Snippet 26, the `grep` command is executed first. The return value of the `grep` command is used by the `if` statement to control the flow of the program.

The `if/ elif/ else` statement is used when one of the conditions can evaluate to `true`. The code in Code Snippet 26 demonstrates the use of `if/ elif/ else` statement.

In Code Snippet 27, the value stored in the variable, `person`, is matched against three constant values, Steve, Todd, and Markus. If the value in the variable matches any one of the three values then the corresponding message is displayed. If it does not match any of the values, then the default message, "Who are you talking about?", is displayed.

#### Code Snippet 27:

```

if [ $person = Steve ]
then
    print $person is on the sixth floor.
elif [ $person = Todd ]

then
    print $person is on the fifth floor.
elif [ $person = Markus ]

then
    print $person is on the fourth floor.
else
    print "Who are you talking about?"
fi

```

### 5.3.2 The case statement

The `case` selection structure compares the value in the variable against every pattern until a match is found. When a match is found, the statements following the matching pattern are executed. If no match is found, the `case` statement exits without performing any action. It also provides a default section that is used if none of the patterns match. It acts as a substitute for multi-line if statement linked with logical `or` symbols (`||`). Each `case` statement is terminated by `esac` statement.

#### Syntax:

```

case variable in
    pattern1 )
        statements ;;
    pattern2 )
        statements ;;
    .
    .
esac

```

The statements corresponding to the first pattern matching the expression are executed, after which the case statement terminates. The variable usually hold the variable's value. The patterns can be plain strings, or they can be expressions using `*`, `?`, `!`, `[]`, etc. (such as file-matching patterns).

In Code Snippet 28, the `case` statement matches the value of the variable `$person` for the strings `steve`, `todd`, and `markus`. The `*` pattern placed as the last option of the `case` statement is selected if the other patterns fail to match. The `"**"` character means everything. In this case it means any other value than what was specified.

#### Code Snippet 28:

```

case $person in
    steve)
        print "He's on the sixth floor." ;;
    todd)
        print "He's on the fifth floor." ;;
    markus)
        print "He's on the fourth floor." ;;
    *)
        print I do not know $person. ;;
esac

```

### 5.3.3 The for- loop repetition statement

A `for` loop can be used to iterate over all items in an array or list and execute commands on each of these items. The `for` loop is terminated by the `done` statement.

#### Syntax:

```

for variable in list-of-values
do
    commands
done

```

The `for` statement continues iterating over the list until the list-of-values is exhausted.

The code in Code Snippet 29 demonstrates the use of the `for` statement. It consists of a list having a series of character strings (a, b, c, d and e) assigned to the variable 'alphabet'. In line 3, the `for` statement loops through each of the letters present in the variable, alphabet. In the body of the loop, the value of the count variable is increased by one and the value of the count variable alongwith the value stored in the variable, letter is displayed.

#### Code Snippet 29:

```

$alphabet="a b c d e"           # Initialize a string
count=0                         # Initialize a counter
for letter in $alphabet
do
    count=`expr $count + 1`      # Increment the counter
    echo "Letter $count is [$letter]" # Display the result
done

```

### 5.3.4 The while-loop repetition statement

The `while` loop uses conditions the same way the `if` statements do. The `while` loop is executed as long as the condition remains true. The code to be executed is written within `do` and `done` statements.

#### Syntax:

```

while condition
do
    commands..
done

```

The code in Code Snippet 30 just replaces the `for` loop set-up in Code Snippet 29 with its equivalent `while` syntax. Instead of stepping through the letters in the variable alphabet, the loop is controlled by monitoring the value of the variable count using the syntax `[ $count -lt 5 ]`. The `-lt` flag represents less-than. The `while` loop is executed as long as the value of the variable count is less than 5. In the body of the loop, the value of the variable, count is incremented and is

used to access the next letter from the variable, `alphabet`. The code uses the `bc` command which represents a precision calculator.

#### Code Snippet 30:

```
alphabet="a b c d e"           # Initialize a string
count=0                       # Initialize a counter
while [ $count -lt 5 ]        # Set up a loop control
do
    count=`expr $count + 1`     # Increment the counter
    position=`bc $count + $count - 1` # Position of next letter

    letter=`echo "$alphabet" | cut -c$position-$position` # Get next letter
    echo "Letter $count is [$letter]" # Display the result
done
```

### 5.3.5 The `continue`, `break`, and `exit` statements

The `break` and `continue` statements are used to interrupt the execution of the loop. The `break` command stops the execution of the current iteration of the loop, and jumps out to the nearest enclosing loop. The `continue` command stops the execution of the current iteration of the loop, and forces the loop to jump to its next iteration.

The code in Code Snippet 31 checks if the value of the variable `index`, is less than or equal to 3. If the result is `true`, it displays "continue" and moves to the next iteration of the `for` loop. When the `for` loop reaches the value 4, the `if` statement returns `false` and the control moves to the next statement following the `if` statement. The values 4 to 7 are then displayed. When the value of `index` in the `for` loop reaches 8, the second `if` condition returns `true`, the `break` statement is executed and the `for` loop terminates. The `-le` operator denotes "less than or equal to" and `-ge` operator denotes "greater than or equal to".

#### Code Snippet 31:

```
for index in 1 2 3 4 5 6 7 8 9 10
do
    if [ $index -le 3 ]; # Checks if index is less than or equal to 3
    then
        echo "continue"
        continue # Moves to the next iteration of the for statement until the index
                  # is less than or equal to 3
    fi
    echo $index
    if [ $index -ge 8 ]; # Checks if index is greater than or equal to 8
    then
        echo "break"
        break # Moves out of the for loop when the index is greater than or equal to 8
    fi
done
```

Sometimes, in the middle of a loop, or somewhere in a script if it may become necessary to exit the execution. This can be achieved using the `exit` command. The user can specify the status of exit to be displayed if there is an abnormal exit. To specify the exit status, the user uses the `exit` command followed by a non-zero number. If no exit status is provided, the `exit` command exits having the status value as zero, which indicates success.

The code in Code Snippet 32 takes two positional arguments. It will exit with status 2 (error) rather than 0 (success) if it is not invoked with two parameters. The `-ne` operator denotes "not equal to".



**Code Snippet 32:**

```

if [ $# -ne 2 ]
# "$#" is number of parameters- here we test
# whether it is not equal to two
then
echo "Usage $0 \<file1\> \<file2\>"      #not two parameters
# so print message
exit 2                                # and fail ($0 is # name of command).
fi

```

**5.3.6 Functions**

Using functions while scripting not only helps to make scripting easier, but also helps in easy maintenance of the code. Functions enable the program to be broken into smaller manageable entities. Each of these entities perform a particular task and can also return a value.

A function always begin with a function name. The commands to be executed are enclosed within the curly braces. The code is reusable. The shell functions must be declared in the shell scripts before being used.

**Syntax:**

```

function-name ( )
{
    command1
    ...
    commandN
    return
}

```

The `function-name` is the name of your function, that executes a series of commands. A `return` statement is used to terminate a function.

The code in Code Snippet 33 defines a function named 'SayHello'. In the function 'SayHello' an `echo` command is used to display a message, "Hello , Have a nice day" on the standard output.

**Code Snippet 33:**

```

SayHello()
{
    echo "Hello , Have a nice day"
}

```

To execute the function, the function name is typed on the command line as shown below.

```

[student@localhost ~]$ SayHello

```

The functions can receive arguments, to make more generic or versatile reusable codes. The code in Code Snippet 34 demonstrates a function to check if the arguments are passed to it.

**Code Snippet 34:**

```
#!/bin/bash

func2 () {
    if [ -z $1 ]
    # Checks if any params.
    then
        echo "No parameters passed to function."
        return 0
    else

echo "Param #1 is $1."
        fi
    }
func2
# Called with no params

func2 first
# Called with one param

func2 first second
# Called with two params

exit 0
```

The function `func2()` checks to see if the number of arguments passed to it is zero. If no arguments are passed, the message "No parameters passed to function" is displayed. If arguments are passed to a function, then the arguments are displayed on the standard output.

The `return` statement in the function can be used to set the value of the variable `$1`. When a value is returned by a function, it is accessible outside the function. The code in Code Snippet 35 demonstrates an example of a function that returns a value.

#### Code Snippet 35:

```
anymore() {
echo "\n$1 ? (y/n) : c" 1>&2
read ans
case "$ans" in
    y|Y) return 0 ;;
    *) return 1 ;;
esac
}
```

The variable `$1` is a special variable that is assigned the value returned from the function. Suppose the function is invoked with the argument string "Want to continue". The string "Want to continue" is stored in the variable `$1`. The user is then prompted to enter 'y/n' i.e. **y** to continue, or an **n** to terminate the outermost loop. If the user enters 'y|Y' the function returns 0. If the user enters 'n' the function returns a 1. To display the value returned by the function, the `echo` command with the argument `$1` is used.

## The Session In Brief

```
#!/bin/bash\nread -p "Want to continue?"\n\nWant to continue?(y/n) : n\n$ echo $1\n1
```

It displays the value returned in the default pattern.

- The shell is an intermediary program that interprets the user-typed commands at the terminal.
- The user can store sequences of frequently used Linux commands in files, called scripts. The shell can read these files and execute the commands within these files.
- The echo and printf commands are used to display some text or the value of a variable. The printf command displays formatted text. The read command is used to read the input from the standard input.
- The if/ elif/ else structure is used when multiple comparison tests have to be performed.
- The case selection structure is used to select a pattern from amongst multiple patterns and execute the statements based on the selected pattern.
- A for loop can be used to iterate over all items in an array or list and execute the statements. The while loops use conditions the same way if statements do; they continue to run until the condition becomes false.
- The break and continue statements are used to interrupt the loop execution.

## Check Your Progress

1. The command to display the formatted output is \_\_\_\_\_.
  - a. output
  - b. echo
  - c. printf
  - d. print
2. The \_\_\_\_\_ command forces the loop to jump to its next iteration.
  - a. loop
  - b. continue
  - c. break
  - d. exit
3. In which of the following structured constructs is the character “\*” used to denote anything when the specified pattern is not present?
  - a. for-loop
  - b. while-loop
  - c. if / else
  - d. case
4. Which of the following are selection structures?
  - a. for-loop
  - b. while-loop
  - c. if / else
  - d. case
5. Which of the following provides the concept of code reusability?
  - a. function
  - b. recursive
  - c. repetition
  - d. command

## Shell Scripting (Lab)

### Session Objectives:

*At the end of this session, you will be able to -*

- Create a basic shell script
- Use the case, if / else, and while structures in shell script
- Use functions in shell script

### Part I - 60 Minutes

---

#### Exercise 1: Creating a script

##### Problem

Give the steps to create a shell script using vi editor.

##### Analysis

The problem requires the user to create a script using the vi editor.

##### Solution

The steps to open the GNOME terminal are as follows:

1. Login as student user. Click Application → Accessories → Terminal from the menu. The Terminal window appears as shown in figure 6.1.

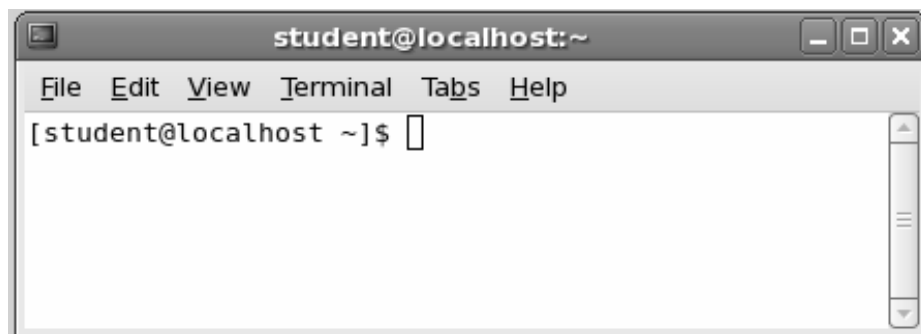


Figure 6.1: Terminal Window

The steps to open vi editor and to create a shell script are as follows:

1. Type the command to create a shell script file as shown in figure 6.2.

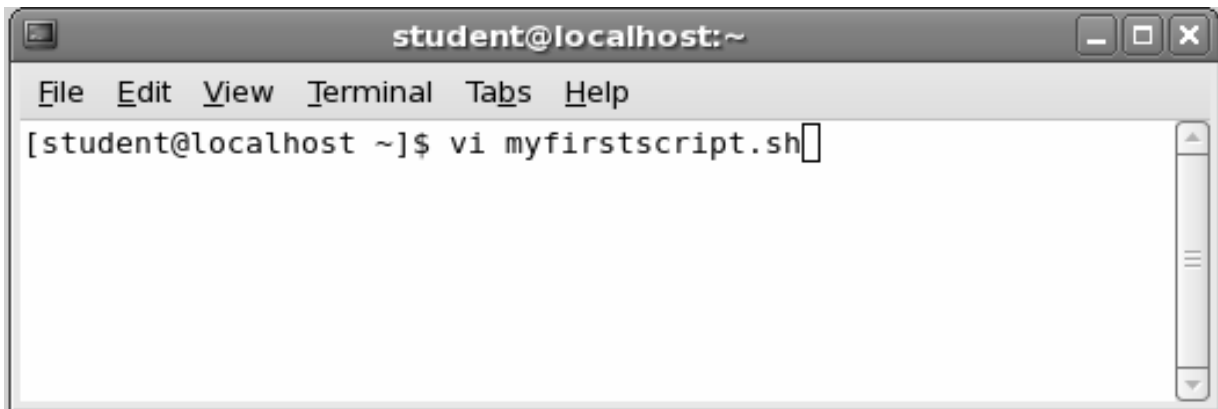


Figure 6.2: Creating a new shell script

2. Press **i** to insert text at the current cursor position.
3. Type the following script as shown in figure 6.3.

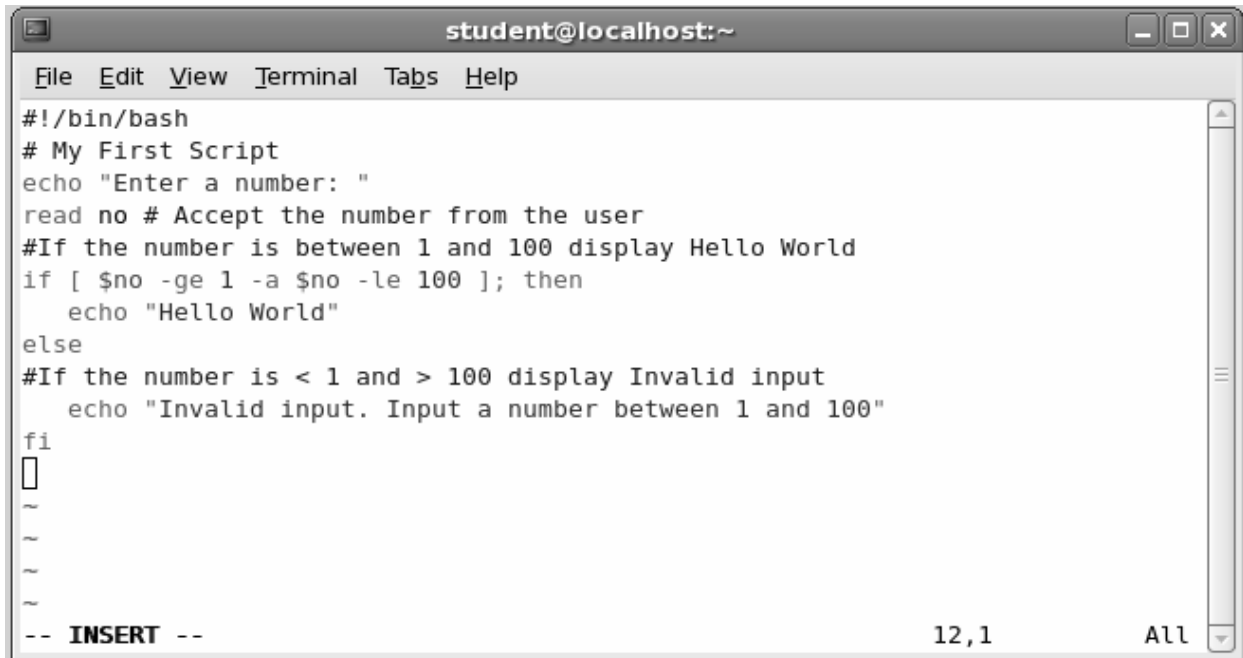
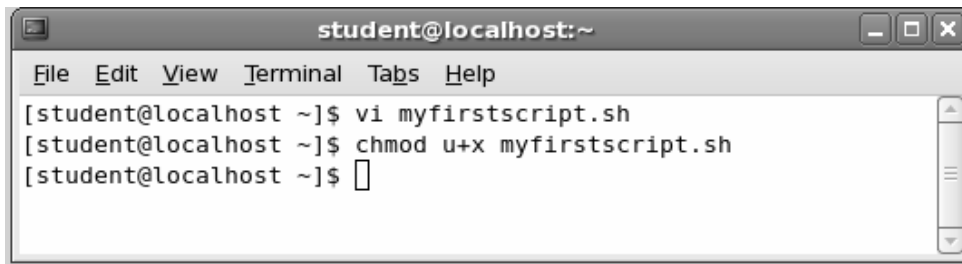


Figure 6.3: myfirstscript.sh

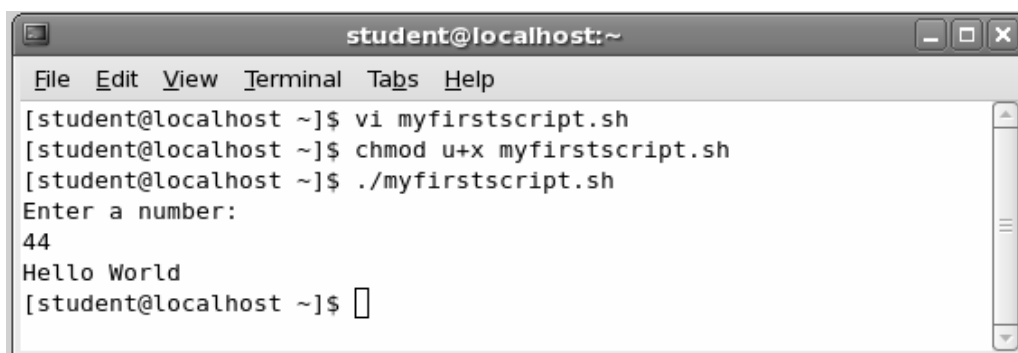
4. Type **:wq**, to save and quit the file.
5. Make the script executable using the **chmod** command as shown in figure 6.4.



```
student@localhost:~  
File Edit View Terminal Tabs Help  
[student@localhost ~]$ vi myfirstscript.sh  
[student@localhost ~]$ chmod u+x myfirstscript.sh  
[student@localhost ~]$
```

**Figure 6.4: Making myfirstscript.sh executable**

The u and x denote the permissions that are assigned to the script file. The 'u' specifies the user who owns the file, and 'x' specifies the permission to execute the file.

**6. Execute the script and enter the details as shown in figure 6.5.**

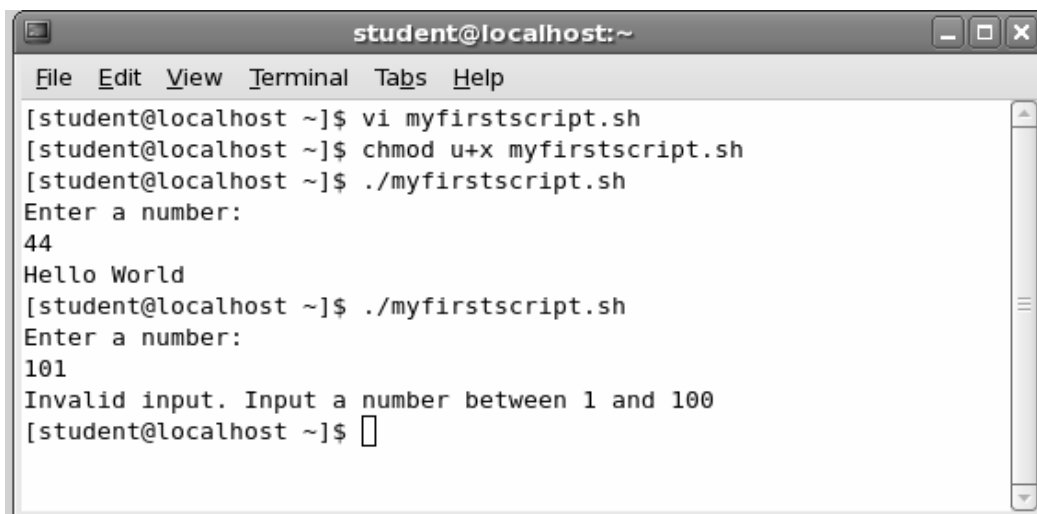
```
student@localhost:~  
File Edit View Terminal Tabs Help  
[student@localhost ~]$ vi myfirstscript.sh  
[student@localhost ~]$ chmod u+x myfirstscript.sh  
[student@localhost ~]$ ./myfirstscript.sh  
Enter a number:  
44  
Hello World  
[student@localhost ~]$
```

**Figure 6.5: Output of myfirstscript.sh**

The user enters the number 44. Since the number 44 is in the range 1 - 100, it displays the output "Hello World".

**7. Execute the script again and enter the number 101.**

The output is shown in Figure 6.6.



```
student@localhost:~  
File Edit View Terminal Tabs Help  
[student@localhost ~]$ vi myfirstscript.sh  
[student@localhost ~]$ chmod u+x myfirstscript.sh  
[student@localhost ~]$ ./myfirstscript.sh  
Enter a number:  
44  
Hello World  
[student@localhost ~]$ ./myfirstscript.sh  
Enter a number:  
101  
Invalid input. Input a number between 1 and 100  
[student@localhost ~]$
```

**Figure 6.6: Output of the script myfirstscript.sh**

The user enters the number 101. Since the number 101 is not in the range 1 - 100, it displays the output "Invalid input. Input a number between 1 and 100".

**Exercise 2: Using while-loop****Problem**

Write a script to print a given number in reverse order. For eg, if no is 123 it must print as 321.

**Analysis**

- ✧ The program requires the user to input a number.
- ✧ Let the input number be stored in a variable named `n`.
- ✧ Declare two variables named `rev` and `sd` to hold the reverse of the number and the remainder respectively.
- ✧ Store the original number in a variable named `orgnum`.
- ✧ Initialize `rev` to "" and `sd` to 0.
- ✧ Find out the remainder by using `n % 10` and store the result in `sd`.
- ✧ Divide the number by 10 and store the result in `n`.
- ✧ Store the previous and the current digits in the variable `rev`.
- ✧ Display the original number and the reversed number.

**Solution**

1. Create the file `reverse.sh` in vi editor and type the statements as shown in figure 6.7.

```

student@localhost:~
File Edit View Terminal Tabs Help
#!/bin/bash
echo "Enter a number"
read n # Accept the number from the user
sd=0 # Store a remainder
rev="" # Store the reverse number
orgnum=$n # Store the original number

# Use while loop to calculate the sum of all digits
while [ $n -gt 0 ]
do
    sd=$((n % 10))
    echo "$sd"
    n=$((n / 10))
    # Store the previous number and the correct digit in rev
    rev=$(echo ${rev}${sd})
done
echo "$orgnum in a reverse order is $rev"
~
~
~
~
~
-- INSERT --
18,1 All

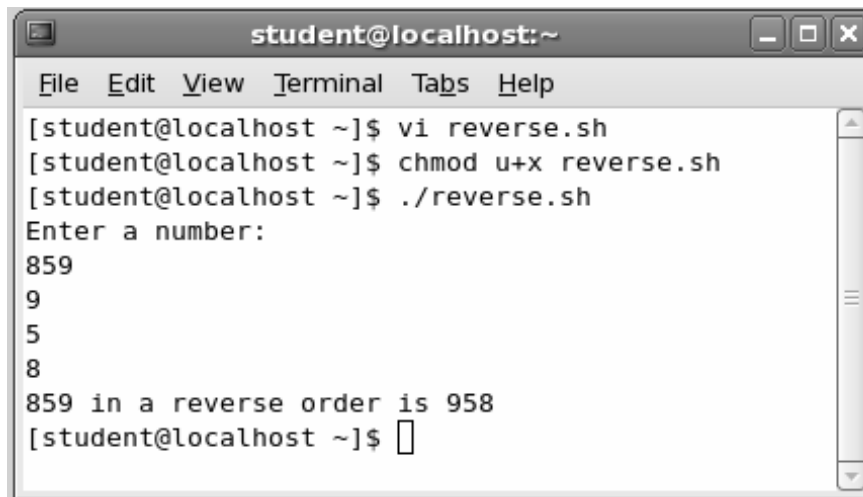
```

Figure 6.7: `reverse.sh`

2. Execute the script.



The output is as shown in figure 6.8.



```

student@localhost:~
File Edit View Terminal Tabs Help
[student@localhost ~]$ vi reverse.sh
[student@localhost ~]$ chmod u+x reverse.sh
[student@localhost ~]$ ./reverse.sh
Enter a number:
859
9
5
8
859 in a reverse order is 958
[student@localhost ~]$

```

Figure 6.8: Output of reverse.sh

### Exercise 3: Using case statement

#### Problem

Write a script that uses the `case` statement to perform the following arithmetic operations:

1. addition (+)
2. subtraction (-)
3. division (/)
4. multiplication (x/ X)

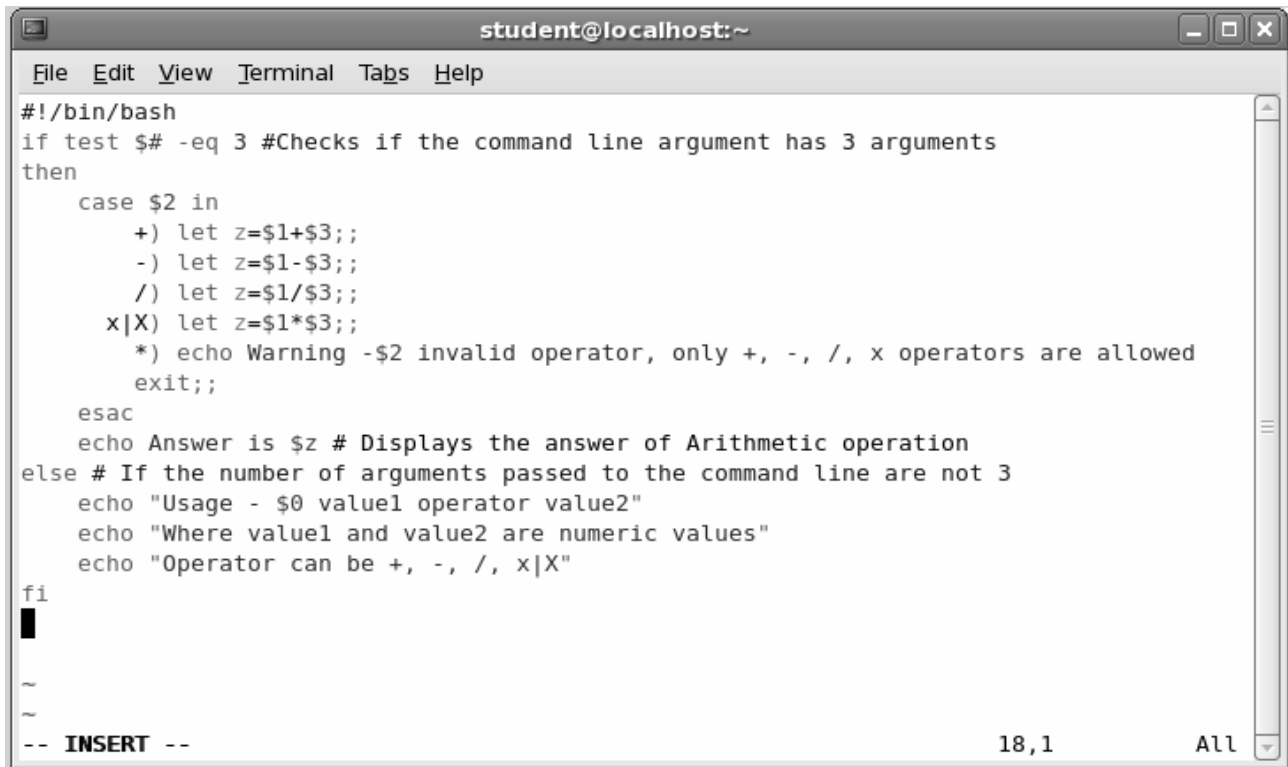
#### Analysis

The input to the program requires two operands and one operator for performing arithmetic operations, that is the total number of arguments passed from the command line is three.

1. Use the `if-then` statement to verify whether three arguments are passed to the script from the command line. If yes, perform the following steps:
2. The second argument passed to the command line is the operator. Match the operator passed against the cases in the `case` structure and perform the appropriate arithmetic operations. The pattern for each of them use the appropriate arithmetic symbols namely, +, -, /, \* respectively.
3. If any other operator is entered, display an error message.
4. If the number of arguments passed to the command line is not 3, it displays a message informing the user about the correct format of the arguments which is, value1 operator value2.

#### Solution

1. Create the file `calculator.sh` in `vi` editor and type the statements as shown in figure 6.9.



```

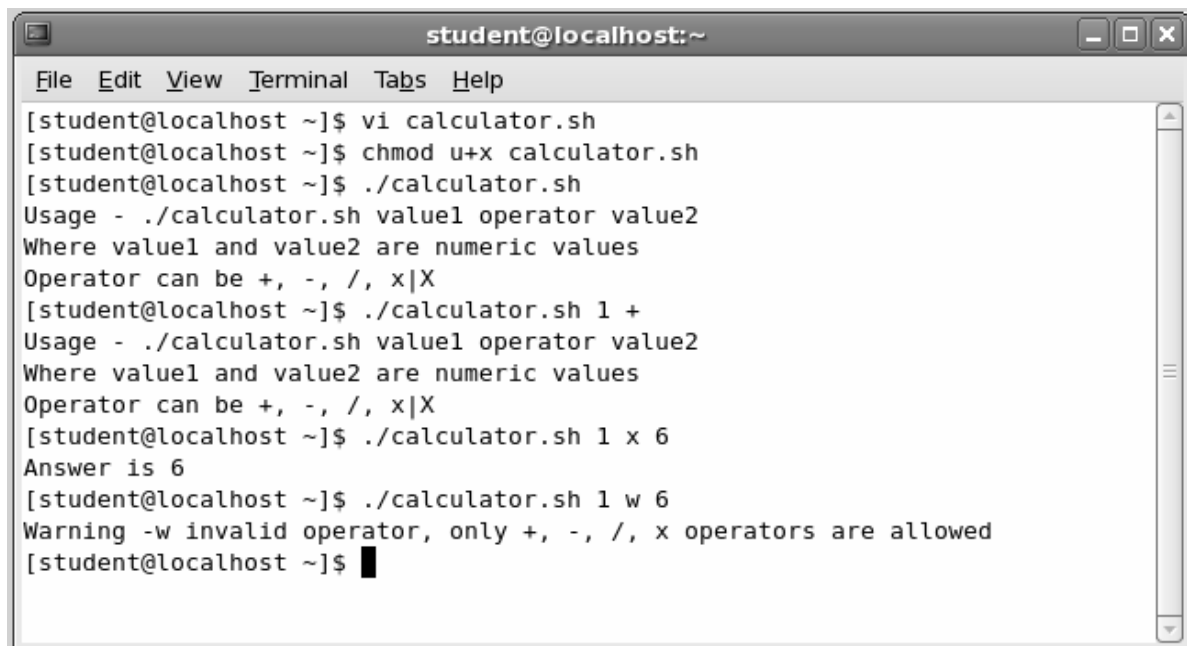
student@localhost:~
File Edit View Terminal Tabs Help
#!/bin/bash
if test $# -eq 3 #Checks if the command line argument has 3 arguments
then
    case $2 in
        +) let z=$1+$3;;
        -) let z=$1-$3;;
        /) let z=$1/$3;;
        x|X) let z=$1*$3;;
        *) echo Warning -$2 invalid operator, only +, -, /, x operators are allowed
           exit;;
    esac
    echo Answer is $z # Displays the answer of Arithmetic operation
else # If the number of arguments passed to the command line are not 3
    echo "Usage - $0 value1 operator value2"
    echo "Where value1 and value2 are numeric values"
    echo "Operator can be +, -, /, x|X"
fi
-- INSERT --
18,1 All

```

Figure 6.9: calculator.sh

## 2. Execute the script.

The output of the various operations is as shown in figure 6.10.



```

student@localhost:~
File Edit View Terminal Tabs Help
[student@localhost ~]$ vi calculator.sh
[student@localhost ~]$ chmod u+x calculator.sh
[student@localhost ~]$ ./calculator.sh
Usage - ./calculator.sh value1 operator value2
Where value1 and value2 are numeric values
Operator can be +, -, /, x|X
[student@localhost ~]$ ./calculator.sh 1 +
Usage - ./calculator.sh value1 operator value2
Where value1 and value2 are numeric values
Operator can be +, -, /, x|X
[student@localhost ~]$ ./calculator.sh 1 x 6
Answer is 6
[student@localhost ~]$ ./calculator.sh 1 w 6
Warning -w invalid operator, only +, -, /, x operators are allowed
[student@localhost ~]$

```

Figure 6.10: Output of calculator.sh

## Exercise 4: Using Functions

### Problem

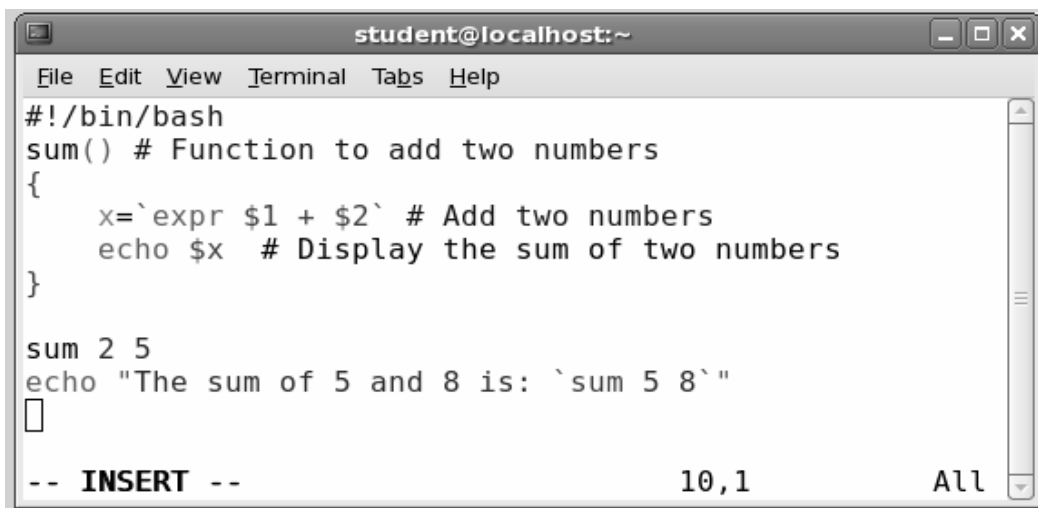
Write a script to add two numbers using functions.

### Analysis

- ✧ The function `sum()` reads the two numbers passed as arguments to it.
- ✧ Create a function named `sum` that contains the logic to add two numbers using `$1 + $2`. The `expr` command is used to perform arithmetic calculation.
- ✧ Call the function in the script below the function definition.
- ✧ Display the message and the output of the addition with the `echo` command.

### Solution

1. Create the file `sum.sh` in vi editor and type the statements as shown in figure 6.11.



```

student@localhost:~
File Edit View Terminal Tabs Help
#!/bin/bash
sum() # Function to add two numbers
{
    x=`expr $1 + $2` # Add two numbers
    echo $x # Display the sum of two numbers
}

sum 2 5
echo "The sum of 5 and 8 is: `sum 5 8`"

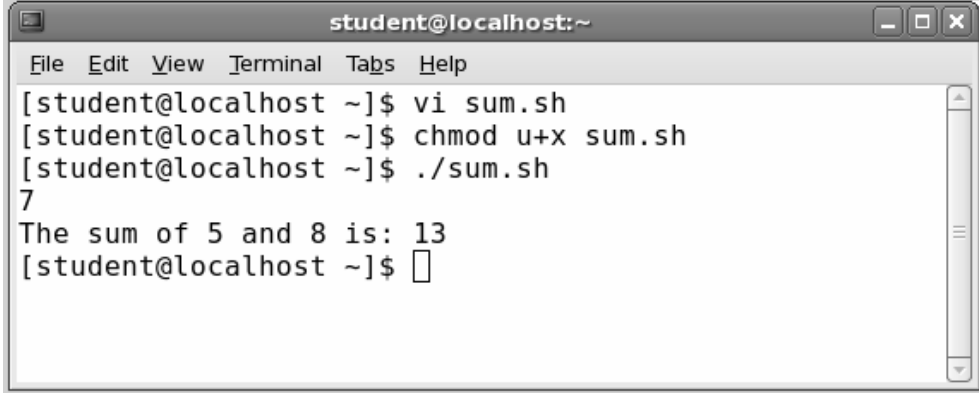
```

Figure 6.11: `sum.sh`

The statement ``sum 2 5`` calls the function `sum` and passes the values 2 and 5 as the arguments to the function. The output is displayed as 7. Then the `sum` function is called again within the `echo` statement by passing the arguments 5 and 8. Within the function, the arguments passed to the function can be accessed in the shell script. `$1` can be used to access the first argument, while `$2` can be used to access the second argument.

2. Execute the script.

The output is shown in Figure 6.12.



```
student@localhost:~  
File Edit View Terminal Tabs Help  
[student@localhost ~]$ vi sum.sh  
[student@localhost ~]$ chmod u+x sum.sh  
[student@localhost ~]$ ./sum.sh  
7  
The sum of 5 and 8 is: 13  
[student@localhost ~]$
```

Figure 6.12: Output of sum.sh

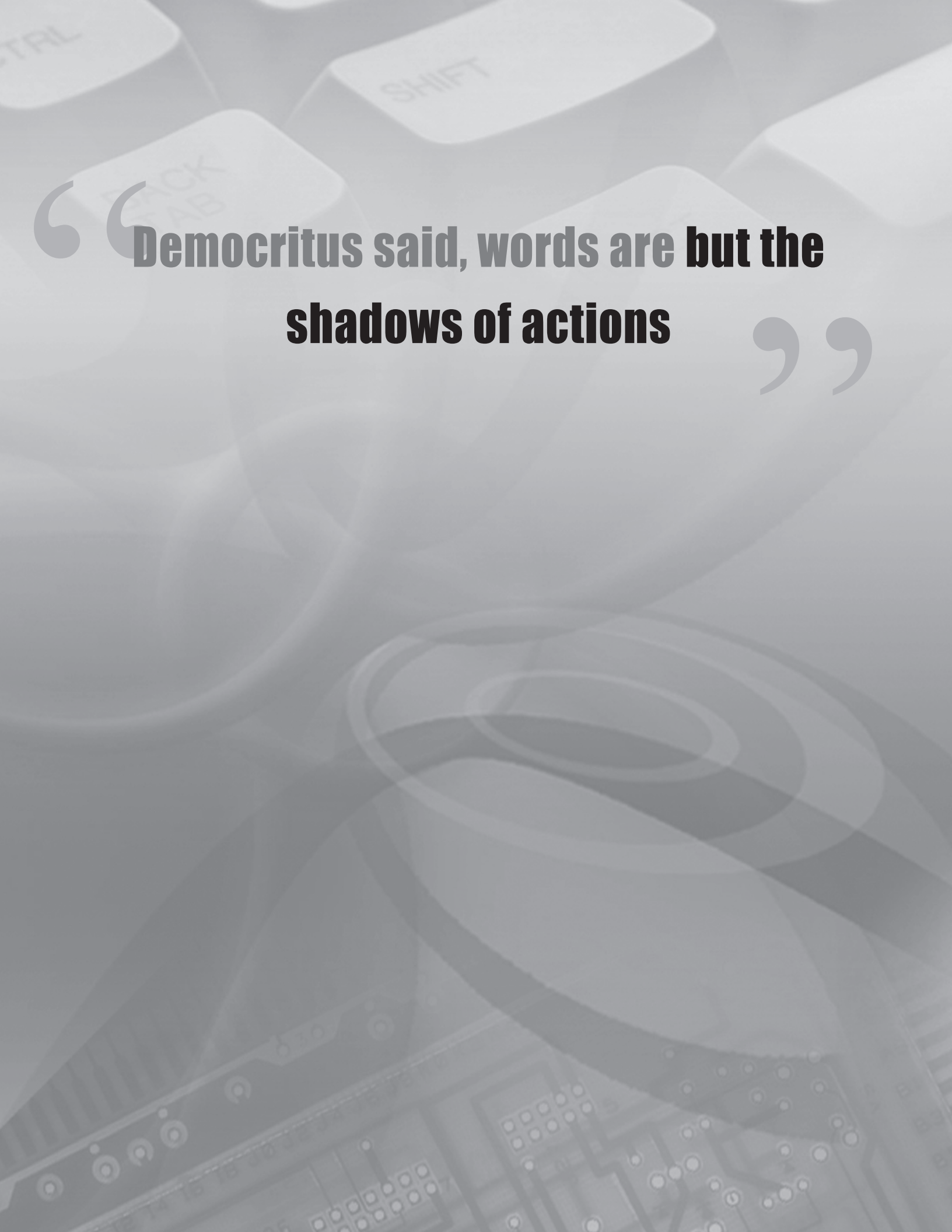
## Part II - 60 Minutes

---

1. Write a shell script to store a name in a variable and then display the name stored in the variable.
2. Write a shell script to accept a number from the user. If the user enters a number between 10 and 50, display the message "The number is between 10 and 50", otherwise display the message "Invalid Entry".  
Hint: Use `if / else` construct.
3. Write a shell script to add the first five even numbers and display the result.

**Do It Yourself**

1. Write a script that accepts any number of command line arguments. Search for the string "is" in the arguments passed to the command line and display the total count.
2. Write a shell script that calls a function to calculate the factorial of a number.

The background is a grayscale, high-contrast image of a computer keyboard and its internal circuitry. The keys are visible in the upper half, with labels like 'CTRL', 'SHIFT', and 'BACK TAB' partially legible. The lower half shows the intricate patterns of a printed circuit board (PCB) with various electronic components and solder points. The overall aesthetic is technical and digital.

**Democritus said, words are but the  
shadows of actions**

# Networking

## Session Objectives:

*At the end of this session, you will be able to -*

- Describe how to configure an Ethernet connection
- Describe the Samba server
- Explain how to manage DNS and Hosts
- Explain how to connect to the internet
- Explain how to use the log files

## 7.1 Configuring the Network

---

Networking can be defined as a collection of two or more computers that are connected with each other for easy sharing of data. Common components used in a basic network include clients and servers, network interface cards, hubs, access points, switches, and routers. A network has a central server that acts as the storage area for files and programs.

A client is a machine that connects to the server for accessing information or resources. The network interface card is a card that is inserted into the computer, so that it can be connected to the network. Hubs are devices that enable a group of users to interconnect with each other. A hub provides a common connection point for the devices in the network. Switches forward the data packets to appropriate ports depending on the recipients the packets are meant for. Routers provide an interface between networks that use different protocols.

A machine requires a network connection to communicate with other machines. This is achieved by configuring the operating system to recognize an interface card, such as Ethernet, ISDN modem, or token ring, and configuring the interface to connect to the network.

The user can use the **Network Configuration** window to configure network interface types. To use the **Network Configuration** window, the user must have the privileges of a superuser or root. The application to configure the network is started by selecting **Network** from the **System Settings**. The user can also access the **Network Configuration** window by typing the command, `"system-config-network"` at the shell prompt.

The **Network Configuration** window is displayed as shown in figure 7.1.

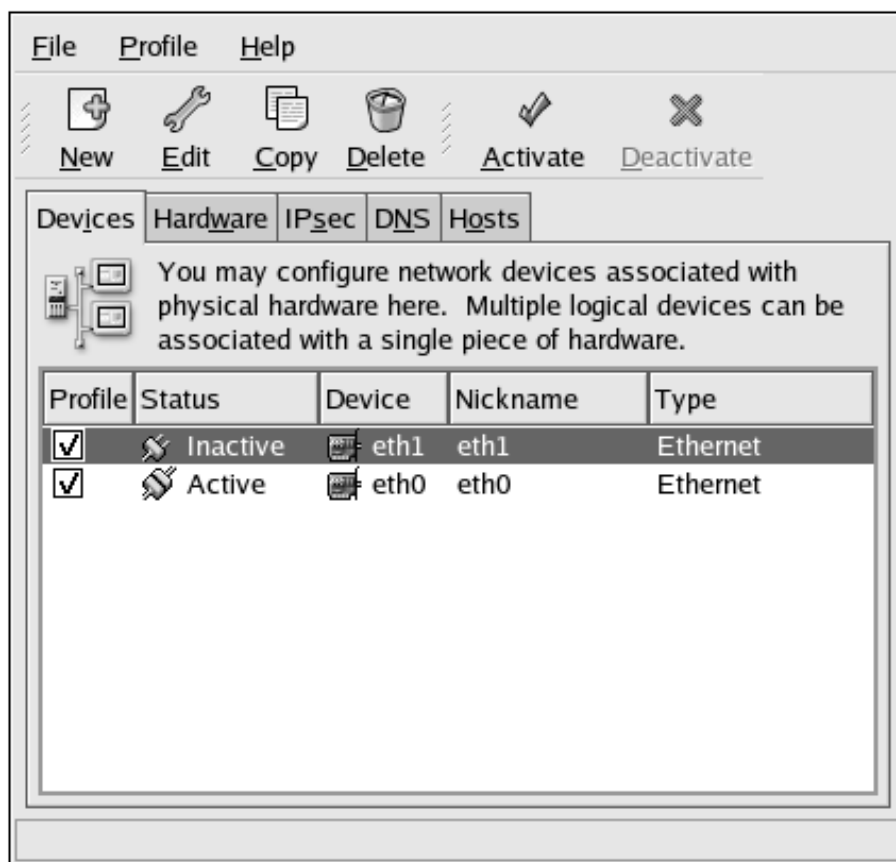


Figure 7.1: Network Configuration window

### 7.1.1 Configuring an Ethernet Connection

To establish an Ethernet connection, the user requires a Network Interface Card (NIC), a network cable, such as the CAT5 cable, and a network for connection. The user must also ensure that the NIC is compatible with the network to which the machine is connected.

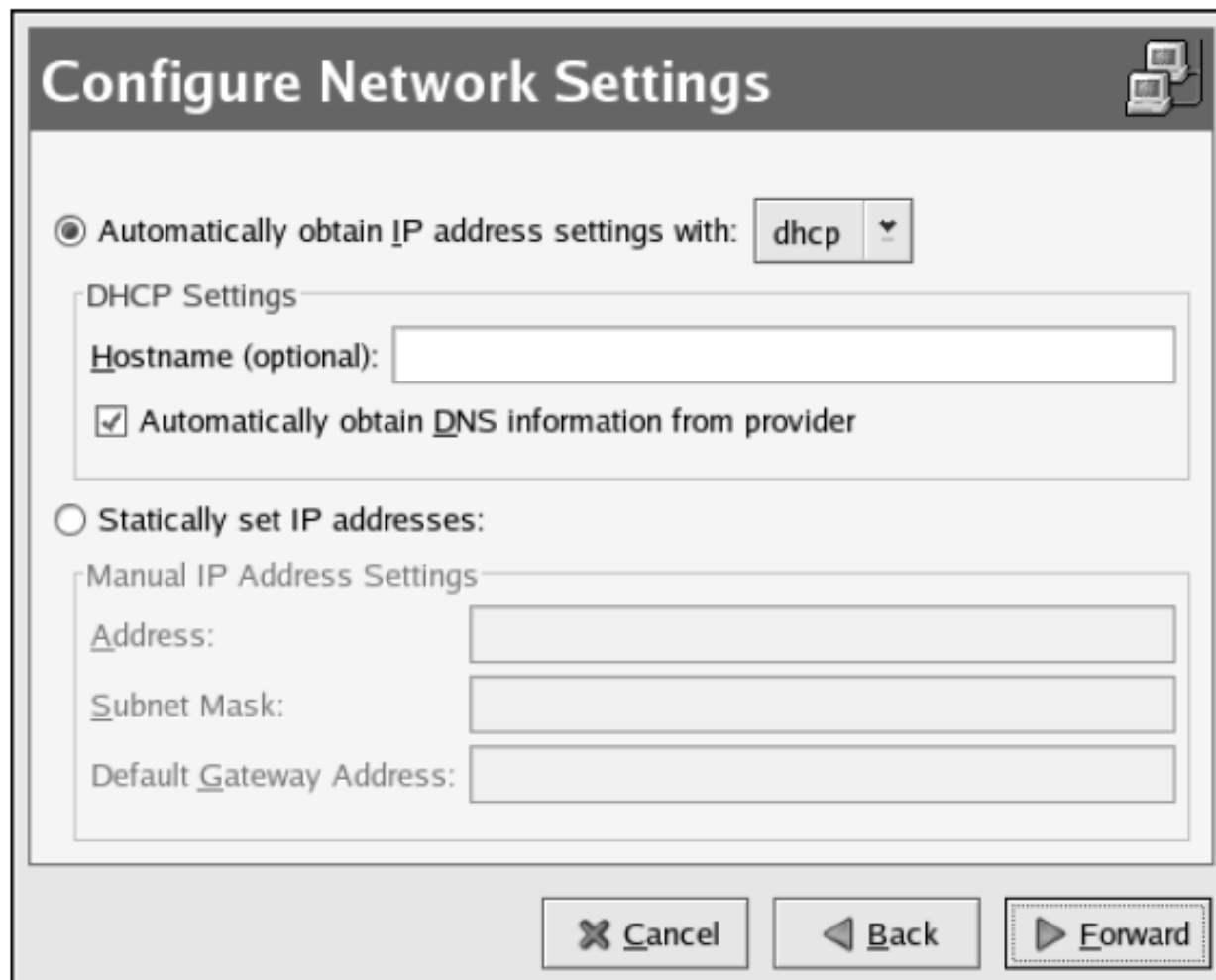
An Ethernet connection can be added, when the **New** button in the **Devices** tab is clicked. The new Ethernet connection is configured by selecting **Ethernet** connection from the Device Type list. If the user has already added the network interface card to the hardware list, the user must select the card from the Ethernet card list. The **Other Ethernet Card** option is used if the user wants to add a new Ethernet card.

During the installation of the Linux Operating System, the Ethernet devices are detected and the user is prompted to configure them. The configured Ethernet devices are displayed in the hardware list under the **Hardware** tab.

If the user selects **Other Ethernet Card**, the **Select Ethernet Adapter** window is displayed. The user must then select the manufacturer and model of the Ethernet card, and the device name. The user must select the device name as **eth0** or **eth1** if it is the first or the second Ethernet Card on the system respectively. The **Network Configuration** window also allows the user to configure the resources for the NIC.

In the **Configure Network Settings** window as shown in figure 7.2, the user must select either the **Dynamic Host Configuration Protocol (DHCP)** or a **Static IP address**. The host name is not specified by the user when the device receives a different IP address each time when the network is started.





The image shows a 'Configure Network Settings' dialog box. At the top, there's a title bar with the text 'Configure Network Settings' and a small icon of two computers. Below the title bar, there are two radio buttons. The first one is selected and is labeled 'Automatically obtain IP address settings with:'. To its right is a dropdown menu showing 'dhcp'. Below this, there's a section titled 'DHCP Settings' which contains a text field for 'Hostname (optional):' and a checked checkbox labeled 'Automatically obtain DNS information from provider'. The second radio button is labeled 'Statically set IP addresses:'. Below it is a section titled 'Manual IP Address Settings' which contains three text fields: 'Address:', 'Subnet Mask:', and 'Default Gateway Address:'. At the bottom of the dialog box, there are three buttons: 'Cancel' (with a close icon), 'Back' (with a left arrow), and 'Forward' (with a right arrow).

## Configure Network Settings

☒ Automatically obtain IP address settings with: dhcp ▾

**DHCP Settings**

Hostname (optional):

☒ Automatically obtain DNS information from provider

☐ Statically set IP addresses:

**Manual IP Address Settings**

Address:

Subnet Mask:

Default Gateway Address:

Figure 7.2: Ethernet Settings

After the user has configured the Ethernet device, the device appears in the device list as shown in figure 7.3.

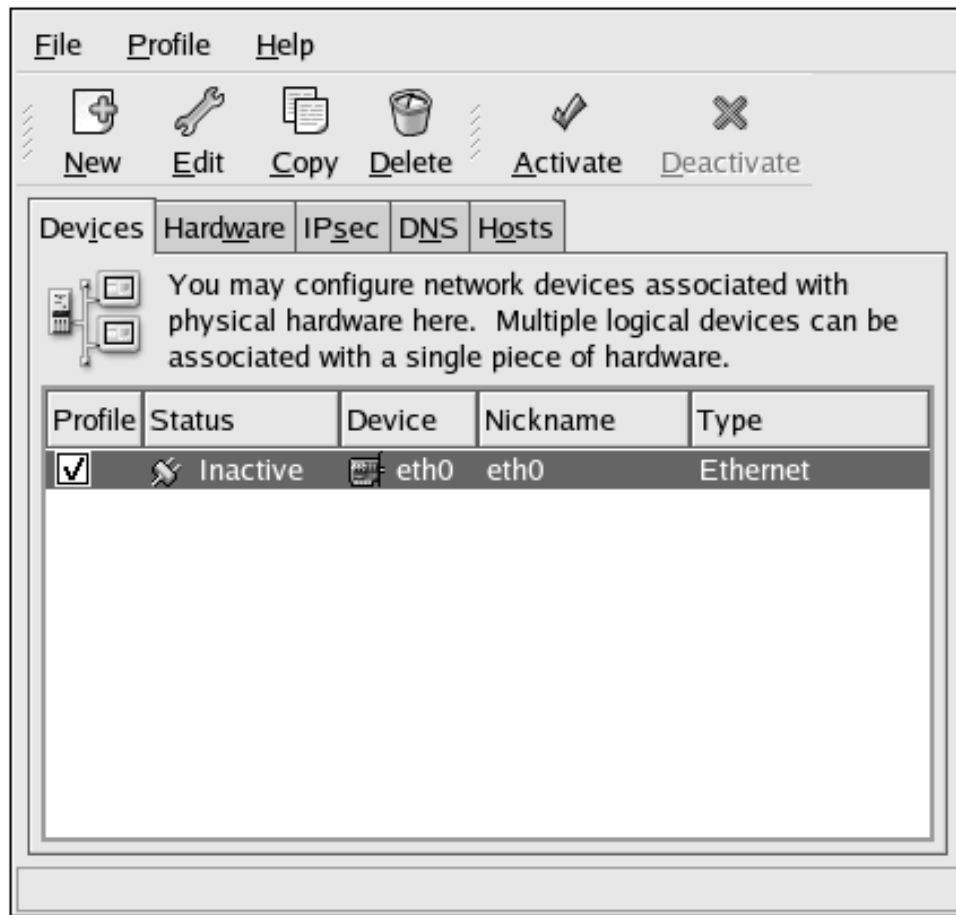


Figure 7.3: Ethernet Device

After adding the Ethernet device, the configuration of the Ethernet device can be edited by selecting the device from the device list and clicking the **Edit** button. For example, suppose that you want to change the settings of an added device, which has been configured to start at boot time by default. For this, the user must select the device to edit it, modify the **Activate** device, and then save the changes.

The device that is added does not get activated by default. To activate the device, the user must select it from the device list, and then click the **Activate** button. If the system is configured to activate the device as soon as the machine starts, the user does not need to perform the step for activation again.

If the user associates more than one device with an Ethernet card, the subsequent devices are device aliases. A device alias allows the user to setup multiple virtual devices for one physical device. Thus, multiple IP addresses can be assigned to a single physical device.

### 7.1.2 Samba Server

Samba is a powerful and versatile server application. Samba uses the Server Message Block (SMB) protocol to share files and printers across a network connection. The user can use Samba when the user has a network that consists of both Windows and Linux machines. Samba allows files and printers to be shared by all the systems in a network. If the user wants to share files amongst Red Hat Linux machines only, he must use NFS. However, if the user wants a sharing of printers between Red Hat Linux machines only, the Samba server is not required.

A default configuration file is provided at the location, `/etc/samba/smb.conf`. This configuration file allows the users to view their Red Hat Linux home directories as a Samba share. This file also allows sharing printers configured for the Red Hat Linux system as Samba shared printers. This sharing of printers allows the user to attach a printer to a Red Hat Linux system and print the files from the Windows machines on the same network.

To configure Samba using a graphical interface, the **Samba Server Configuration** window is used. The **Samba Server Configuration** Window is a graphical interface for managing Samba shares, users, and basic server settings. This window modifies the configuration files in the `/etc/samba/` directory. Any changes to these files that are not made are preserved.

To use this configuration window, the user must run the X Window System, have root privileges, and have the `system-config-samba` RPM package installed. The **Samba Server Configuration** window is invoked by selecting the **Samba Server** from the **Server Settings** in the **System Settings**. Alternatively, the user can also type the command `system-config-samba` at the shell prompt. The **Samba Server Configuration** window is appears as shown in figure 7.4.

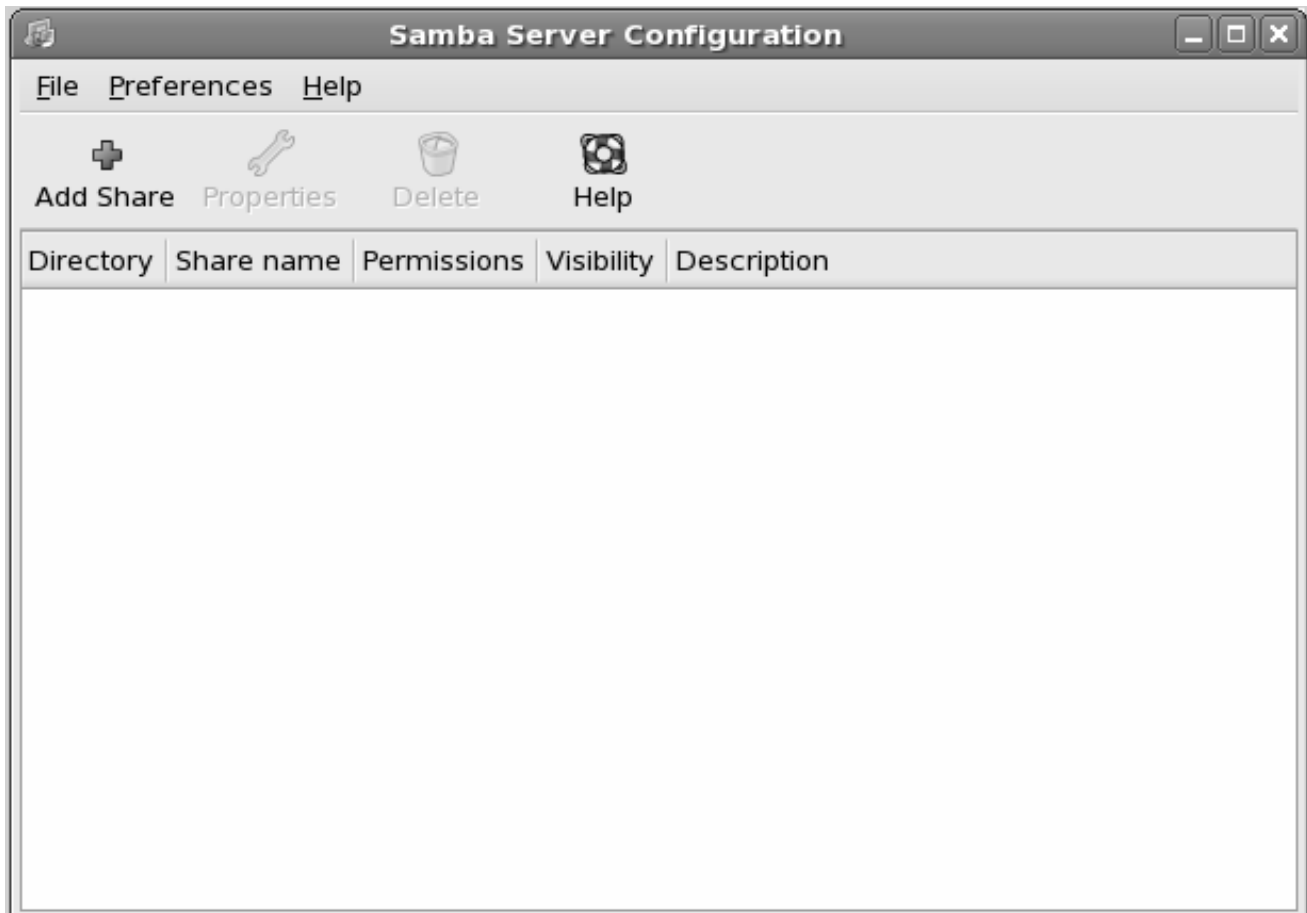


Figure 7.4: Samba Server Configuration window

The **Samba Server Configuration** window does not display shared printers or the default statement allowing users to view their own home directories on the Samba server. The basic settings for the server and some security options need to be configured for configuring a Samba server. The basic settings in the Samba server can be configured by selecting **Server Settings** from the **Preferences** menu within the **Samba Server Configuration** window. The **Basic** tab with default values of the basic settings are displayed as shown in Figure 7.5.



Figure 7.5: Configuring Basic Server Settings

The workgroup that the computer must belong to and a brief description of the machine can be specified in the **Basic** tab. These entries in the **Basic** tab correspond to the `workgroup` and `server` string options in `smb.conf` file. The **Security** tab stores the security-related information pertaining to the server as shown in figure 7.6.



Figure 7.6: Configuring Security Server Settings

The various options available in the **Security** tab are as follows:

- **Authentication Mode:** This corresponds to the security option. The different options that are displayed are ADS, Domain, Server, Share, and User.
  - ✧ **ADS:** For the Samba server to act as a domain member in an **Active Directory Domain** (ADS) domain, the kerberos should be installed and configured on the server. Also, the Samba Server should be a member of the ADS domain. This can be achieved using the `net` utility which is a part of the `samba-client` package. The domain of the Kerberos server is specified in the **Kerberos Realm** field.
  - ✧ **Domain:** To verify the user, the Samba server depends on Windows NT Primary or Backup Domain Controller. The username and password are sent by the server to the Controller. The **Authentication Server** field is used to specify the NetBIOS name of the Primary or Backup Domain Controller. The **Encrypted Passwords** option must

be set to **Yes** if this is selected.

- ✧ **Server:** It helps in verifying the username and password which are passed by the Samba server.
- ✧ **Share:** Signifies that the user is prompted for a username and password only when they try to connect to a specific shared directory from a Samba server.
- ✧ **User:** Signifies if the user wants to use the Windows username.
- **Encrypt Passwords:** The user must enable this option if the clients are connecting from a Windows 98, Windows NT 4.0 with service pack 3, or other recent versions of Microsoft Windows. The passwords are exchanged between the server and the client in an encrypted form and not just as plain-text.
- **Guest Account:** The users or guest users are mapped to a valid user on the server as soon as they log into a Samba server.

The changes made in the **Security** tab are then written to the configuration file.

### **7.1.3 Managing DNS and Hosts**

The IP address is a unique address that identifies all the machines in the network. It is not easy to remember all the computers by their IP addresses, so every computer is assigned a unique name for identification. The Domain Name Service (DNS) or name server is an effective way to configure such name based connections.

The DNS enables the user to locate a computer using its name. DNS resolves the host names provided by the users into their respective IP addresses when the users transact with the computer. Conversely, DNS can provide the fully qualified domain name of the computer when its IP address is provided.

To add a host:

1. Open the **Network Configuration** window.
2. Click the **DNS** tab to display the details as shown in figure 7.7.

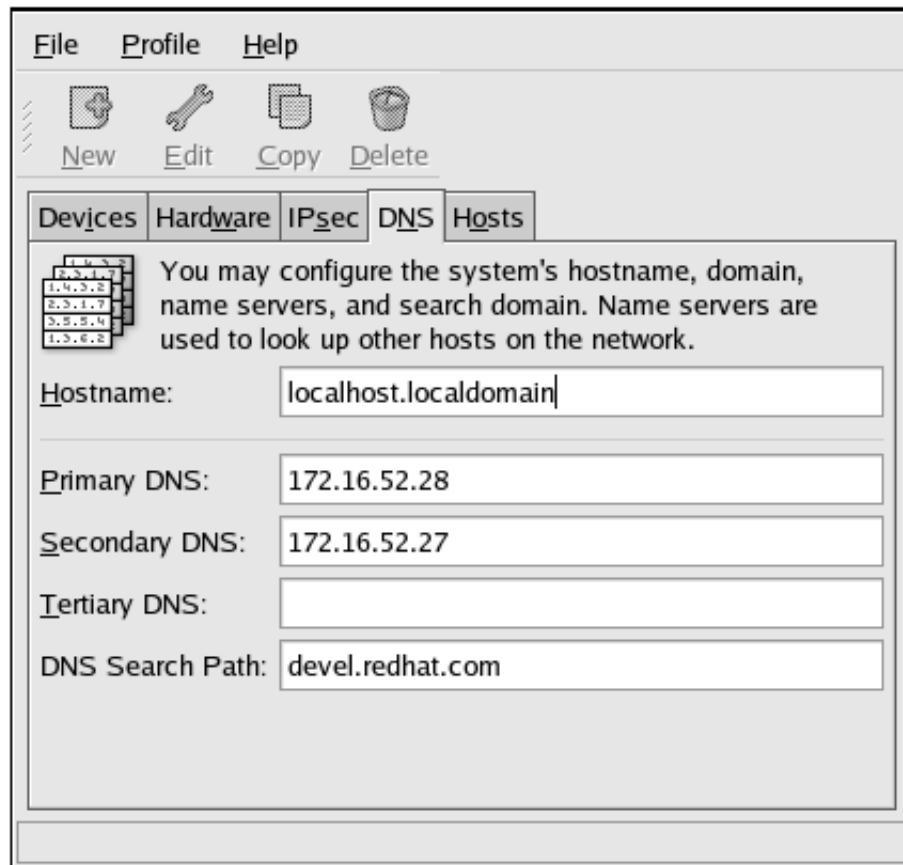


Figure 7.7: DNS Tab

The Primary and Secondary DNS details are entered.

3. Click the **Hosts** tab.
4. The **Hosts** tab appears as shown in figure 7.8.

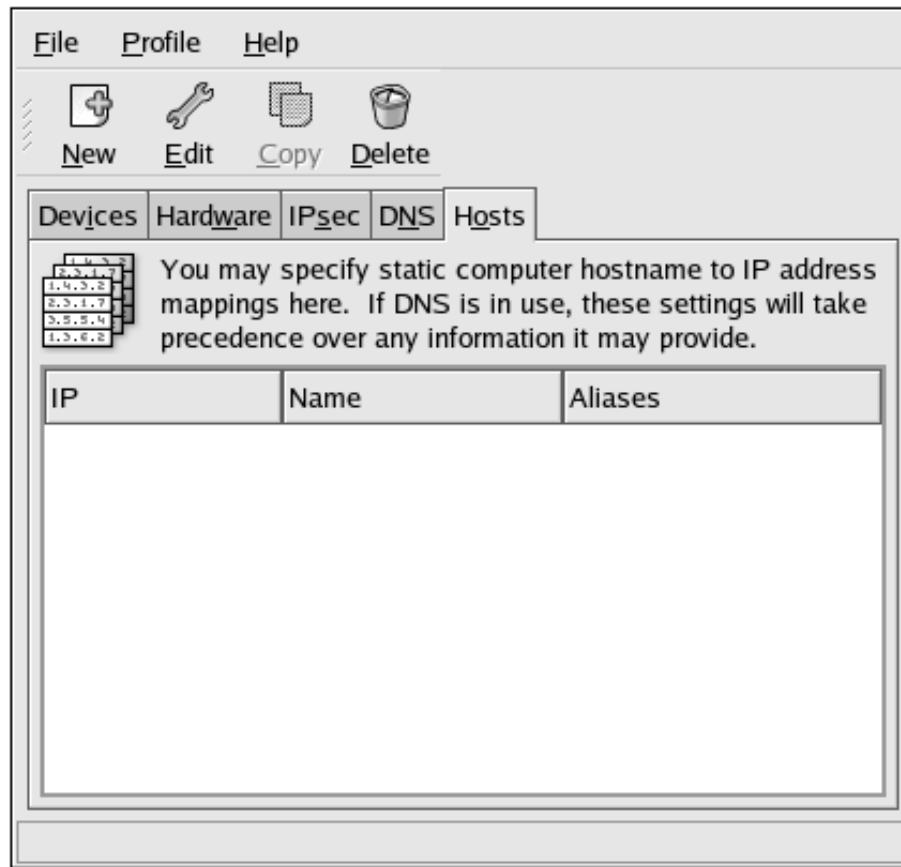


Figure 7.8: Hosts Tab

The **Hosts** tab enables the user to specify the list of IP addresses that act as hosts for the DNS server. The IP addresses and their corresponding hostnames are stored in the **/etc/hosts** file. To resolve a hostname to an IP address, the system refers to the **/etc/hosts** file before using the name servers.

5. Click **New**. The **Add / Edit Hosts entry** dialog box appears as shown in figure 7.9.
6. The IP address, host name, and aliases for the host are specified in this window.

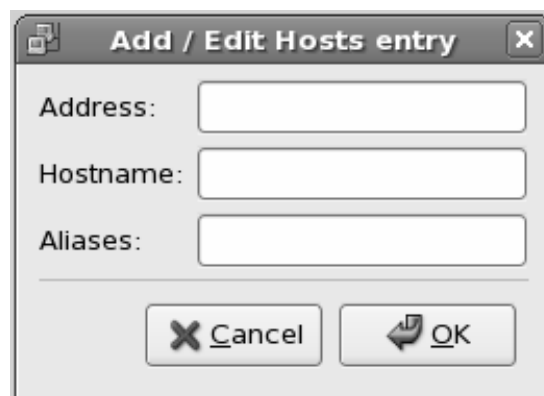


Figure 7.9: Add/Edit Hosts screen

7. Click **OK** to add a new host.
8. Save the file.
9. The **Information Dialog** box appears.

10. Click **OK**.

### 7.1.4 Internet Connection

The key feature of the Linux system is that it is a great platform for offering network services. E-mail is an important feature of any Web site that the user creates. Some of the e-mail protocols used nowadays include SMTP (Simple Mail Transfer Protocol), POP (Post Office Protocol), IMAP (Internet Message Access Protocol), and Dovecot.

- **SMTP:** It is a TCP/IP protocol used in sending and receiving e-mail. This protocol is essentially the communications language that the MTA's (Mail Transport Agent) use to talk to each other and transfer the messages back and forth. The user can configure the SMTP server in a local machine to handle the mail delivery. The user can also configure the remote SMTP servers for outgoing mails. The key feature of SMTP is it does not require authentication. By default, Sendmail is the default SMTP program.
- **POP:** This is used to retrieve e-mails from a remote server over a TCP/IP connection. The e-mail messages are downloaded by the e-mail client applications. It is an Internet standard that is fully compatible with MIME (Multipurpose Internet Mail Extensions). Security can be added by the use of Secure Socket Layer (SSL) encryption for client authentication and data transfer sessions.
- **IMAP:** It is a standard protocol for accessing e-mail from your local server.
- **Dovecot:** This is configured to use IMAP and POP. Implicitly, dovecot runs only IMAP.
- **E-mail program classifications:** E-mail applications fall into one of the following categories namely, Mail Transport Agent (MTA), Mail Delivery Agent (MDA) and Mail User Agent (MUA). An e-mail is a message created using a mail client program. The created message is then forwarded to the server. The server then forwards the mail to the recipient's e-mail client.
  - ✧ **MTA:** It is the program responsible for receiving incoming e-mails and delivering the messages to individual users.
  - ✧ **MDA:** It is invoked by the MTA to store the incoming e-mail in the proper user's mailbox.
  - ✧ **MUA:** It is the software that allows a user to access and manage e-mail, including reading, composing, disposing, printing and displaying e-mail messages. The MUA provides the interface between the user and the MTA. Eudora and Outlook are two popular MUAs.

The steps in the mail delivery process include:

1. The user composes the message and decides to send it. The user's mail agent forwards the message to a central mail gateway.
  2. The gateway reads the message and extracts the destination address from it.
  3. After verifying the e-mail address of the destination, MTA will notify MUA that the mail was sent.
  4. The MTA will now deliver the message to the configured mail exchange server for each domain. If no mail exchangers are available, the message is put in a queue to be delivered it later.
  5. When the message reaches the final destination, it is handed over to the system MDA by the target MTA.
  6. The MDA stores the message in a spool file, or passes it through filters, or performs other instructions as required.
  7. The user can then retrieve their mail either locally by reading from a spool file, or remotely, by using a protocol such as POP or IMAP.
- **Types of Mail Transfer Agents:** One of the most commonly used MTAs is Sendmail. Other MTAs provided by Red Hat Enterprise Linux are Postfix and Exim.
    - ✧ **Sendmail:** It is a mail transfer agent that enables the user to send e-mail messages to different hosts. Sendmail



communicates with the POP3 or the IMAP protocol for sending or receiving messages.

Some of the options that can be used to determine sendmail status are shown in table 7.1.

Option	Description
sendmail -bp or mailq	Displays the status of the mail queue. List e-mail files waiting to be sent or incoming
ps ax grep sendmail	Lists the process status of the sendmail daemon
sendmail -bi or newaliases	Rebuilds the aliases database

**Table 7.1: Options of the sendmail**

- ✧ **Sendmail configuration file** : Most of the parameters to control how sendmail works are stored in the `/etc/sendmail.cf` file.
- ✧ **Start/Stop sendmail**: Whenever changes are made to the `sendmail.cf` file, sendmail must be stopped and restarted before the changes take effect. Following are the steps to be performed to do this:

```
cd /etc/rc.d
./sendmail stop
./sendmail start
```

- ✧ If the connection to the Internet is via a dialup line, use the command `fetchmail` with sendmail to get e-mail from the ISP's server. Generally, the use of the `fetchmail` command is to check if any new e-mail exists on the ISP server.

`fetchmail -c` displays the number of messages waiting to be retrieved.

- ✧ The locations where the e-mails are stored are listed in table 7.2.

Location	Purpose
/var/spool/mail	Stores the received mail messages in the directory
/var/mqueue	Specifies the mails waiting to be sent
/var/log/mail	Specifies a log of mail sent and received
/etc/mail	Specifies mail configuration files

**Table 7.2: Location of e-mail storage**

- ✧ **Standard e-mail clients**: The common e-mail clients such as Eudora, Outlook, Netscape are used to send and read e-mail. A Linux `mail` command can also be used to send and read e-mail. The mail command is mostly used for testing e-mail as its not a very user-friendly utility. Some of the `mail` commands are listed in table 7.3.

E-mail Client	Purpose
mail	Reads e-mail. Displays a summary list of unread e-mail messages
mail james	Creates an e-mail message to the local user 'james'. Prompts to enter the 'subject' and then the body of the message. The user needs to end the message with a period. This exits the mail and ensures that the message is sent.
mail james@aptech.com	Specifies that if the domain is aptech.com, this will send e-mail to the external user

**Table 7.3: Standard e-mail clients**

An alternative to `mail` is the e-mail client's `pine` command. The `pine` utility is usually installed in Linux. It is a user friendly utility. For accessing this utility, the user needs to type the command `pine` at the command prompt.

- ✧ **Postfix:** This is an alternative e-mail agent to send mail based on the Simple Mail Transfer Protocol. Postfix has high speed and security. It is compatible with sendmail. It provides easy administration.

The `mail` command is used to send an e-mail from the shell prompt.

### Syntax:

```
[student@localhost ~]$ mail [options] [ -m message_type ] recipient ...
```

The common options available with the `mail` command are summarized in table 7.4.

Option	Function
-t	A 'To:' line is added to the message header for each recipient.
-w	A mail is sent to a remote recipient without waiting for the completion of the remote transfer program.
-m	A Message-Type: line is added to the message header with the value of message_type.
-e	Mail is not printed
-h	A window of headers is initially displayed rather than the latest message. The display is followed by the ? prompt
-p	All the messages are printed
-r	Messages are printed in first-in, first-out order
-f file	Mail uses file (such as mbox) instead of the default mailfile
-s	Specify subject on command line

**Table 7.4: Options of the mail command**

Consider an example to send an e-mail to davidb@aptech.com as shown in Code Snippet 1.

### Code Snippet 1:

```
[student@localhost ~]$ mail davidb@aptech.com
Subject: Hello
Hi,
Attend a meeting tomorrow at 9:00
Regards
.
Cc:
```

The user needs to type a '.' (period) at the end of the mail to send the e-mail.

The code in Code Snippet 2 attaches the file (/tmp/message in this case) to the mail body and sends the mail.

### Code Snippet 2:

```
[student@localhost ~]$ mail -s 'Hi' davidb@aptech.com < /tmp/message>
```

The user can also access the e-mail when the GUI is not available. The `mutt` is an e-mail client that manages e-mail in a text-based environment. When there are a series of related messages, such as the original message, followed by the replies to that message, the `mutt` displays each reply under the parent. Thus, `mutt` has the capability of handling threads and is a flexible e-mail client. These messages which are threads can be navigated, deleted and marked as read or unread as a group.

The user can read one mailbox at a time using the `mutt`. To access a specific mailbox, the `-f` option is used along with

the `mutt`. The code in Code Snippet 3 reads a mailbox in an IMAP account.

#### Code Snippet 3:

```
[student@localhost ~]$ mutt -f imaps://username@host.com
```

## 7.2 Log Files

Linux stores the messages about the system, including the kernel, services and applications running on it in a file known as a Log file. There are different log files that store different information. For example, a default system log file is a log file used for storing security messages. Log files are useful in troubleshooting a problem with the system such as attempting to load a kernel driver or unauthorized log in attempts to the system.

### 7.2.1 Locating Log Files

System logs are automatically created by the user at the time of installation of various packages. The system logs enable verification of the installed packages, record any errors encountered during booting, and automate tasks. Cron is a utility that can help in automating tasks in Linux.

The two services **syslogd** and **klogd** are responsible for maintaining system and kernel logs. **Syslogd** records messages related to services to programs and applications. **Klogd** records kernel messages. **Klogd** also sends most messages to the **syslogd** facility but occasionally pops up messages at the console. **Syslogd** handles the task of processing most of the messages and sending them to the appropriate file or device. This is configured within **/etc/syslog.conf**.

Generally, the log files are located in the **/var/log/** directory. Individual programs can handle their own logs. The logs for the programs are usually stored within **/var/log/program-name/**. There are many files in the log file directory with numbers appended to them. These files are created when the log files are rotated. Rotation of the log files is performed to reduce the size of the file. The log files are automatically rotated by a **logrotate** package having a cron task according to the **/etc/logrotate.conf** configuration file and the configuration files is stored in the **/etc/logrotate.d/** directory.

### 7.2.2 Viewing Log Files

The log files which are in plain text format can be viewed using text editors such as `vi`. The **System Log Viewer** is used to view the log file in an interactive, real-time application. The **System Log Viewer** is started by selecting the **System Logs** from the **System** menu. The user can also start the **System Log Viewer** by typing the command `gnome-system-log` at the shell prompt. The **System Log Viewer** screen appears as shown in figure 7.10.

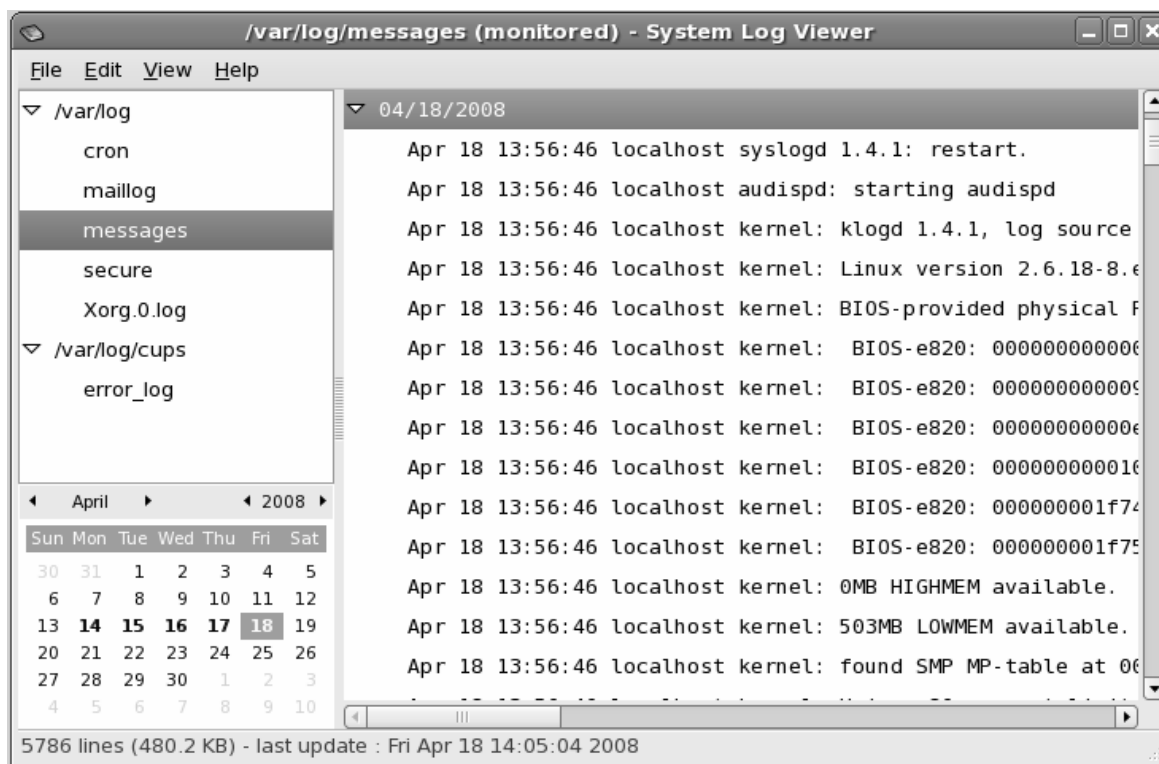


Figure 7.10: System Log

The user can also filter the contents of the selected log file, by selecting **Filter** from the **View** menu. Selecting the **Filter** menu item will display the **Filter** text field, using which the user can specify the filter criteria. The **Clear** button is used to clear the filter.

### 7.2.3 Adding a Log File

Selecting **Open** from the **File** menu helps in adding a log file. This will display the **Open Log** window, using which the user can select the directory and filename of the log file to view as shown in figure 7.11.

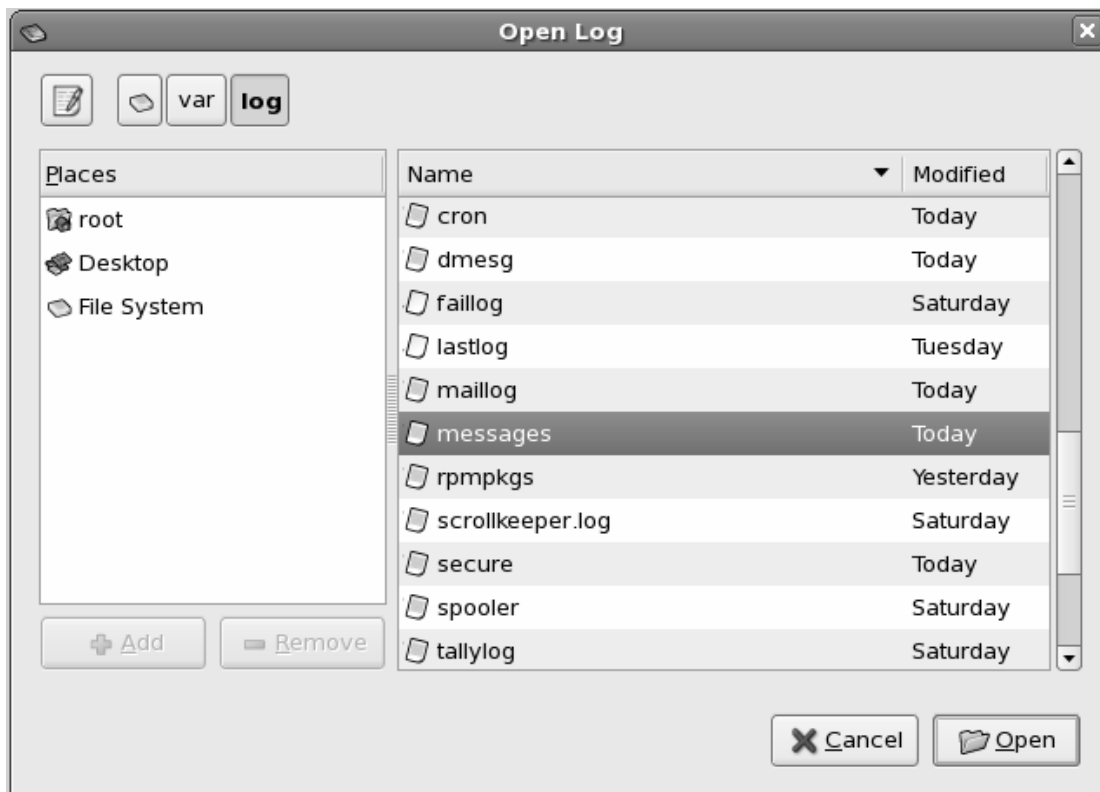


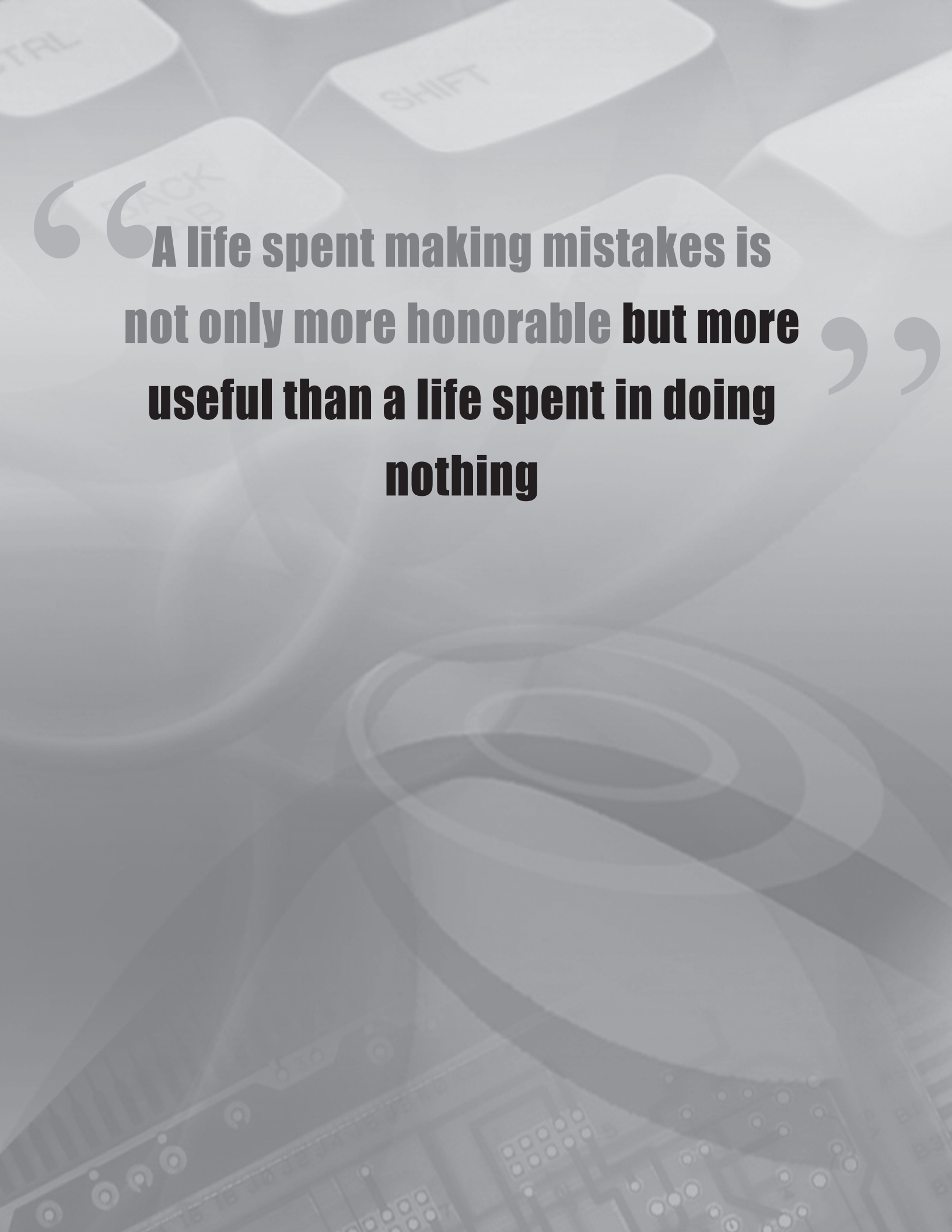
Figure 7.11: Adding a Log File

## The Session In Brief

- A network interface provides a connection between the user terminal and the network.
- To establish an Ethernet connection, the user requires a Network Interface Card (NIC), a network cable such as the CAT5 cable, and a network for connection.
- Samba uses the Server Message Block (SMB) protocol to share files and printers across a network connection. The user can use Samba when the user has a network that consists of both Windows and Linux machines.
- Sendmail is a mail transfer agent that enables the user to send e-mail messages to different hosts.
- To send an e-mail from the console, the mail command can be used.
- Postfix is an alternative e-mail agent to sendmail based on SMTP.
- To view system log files in an interactive, real-time application, the user must use the Log Viewer.

## Check Your Progress

1. You can use the \_\_\_\_\_ to configure a network interface type in Linux.
  - a. The Network Administration window
  - b. Samba Server Configuration window
  - c. Network Interface
  - d. Log Viewer
2. The \_\_\_\_\_ element of the Security tab is used to allow a guest user to log into a Samba Server.
  - a. Encrypt Password
  - b. Guest Account
  - c. Authentication Mode
  - d. Authorization Mode
3. Which of the following interfaces are configured using the Network Administration window?
  - a. Ethernet
  - b. ISDN
  - c. Modem
  - d. Protocol
4. You want to configure the `sendmail` client, which file will you edit?
  - a. `/etc/sendmail`
  - b. `/etc/mail/sendmail.cf`
  - c. `/etc/mail/sendmail`
  - d. `/etc`
5. \_\_\_\_\_ is a computer program or software agent that transfers electronic mail messages from one computer to another.
  - a. Mail Transport Agent
  - b. Mail User Agent
  - c. Mail Delivery Agent
  - d. Mail Server Agent



**“A life spent making mistakes is  
not only more honorable but more  
useful than a life spent in doing  
nothing”**



## Networking (Lab)

### Session Objectives:

*At the end of this session, you will be able to -*

- Configure an Ethernet connection
- Configure Samba server
- View log files

### Part I - 60 Minutes

---

#### Exercise 1: Configure an Ethernet Connection

##### Problem

Configure the Ethernet connection.

##### Analysis

After inserting network cards, you can configure the appropriate interfaces or create a new interface. The new interface card is configured from the **Network Configuration Tool** dialog box.

##### Solution

To create a network interface:

1. **Select System → Administration → Network.**

The **Network Configuration** screen appears as shown in figure 8.1.

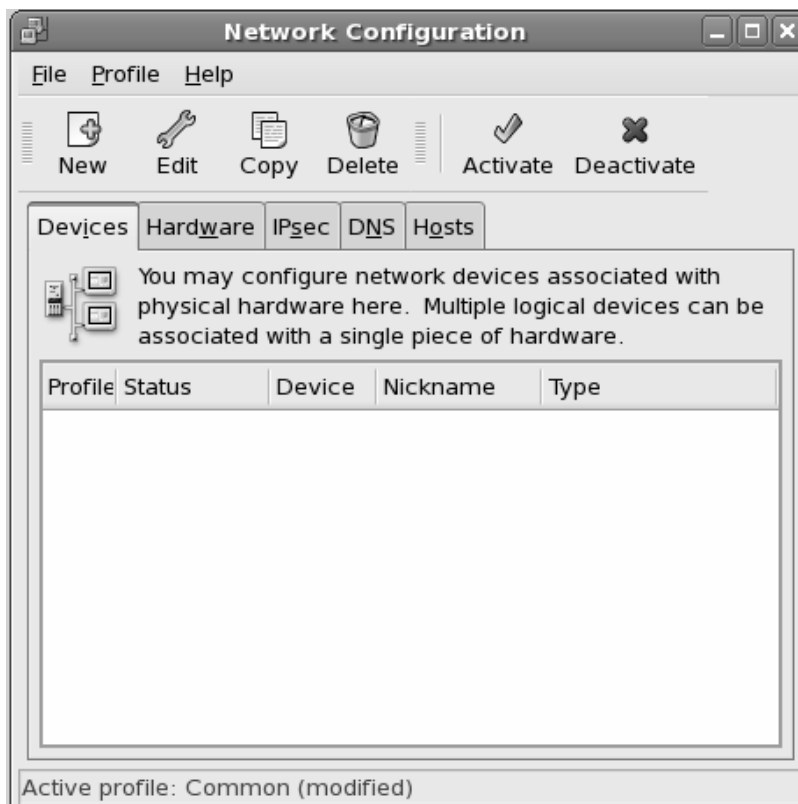


Figure 8.1: Network Configuration screen

**2. Click New.**

The **Add new Device Type** screen appears as shown in figure 8.2.



Figure 8.2: Add new Device Type screen

3. Select Ethernet connection from the Device Type list.
4. Click Forward.

The **Select Ethernet Device** screen appears as shown in figure 8.3.



Figure 8.3: Select Ethernet Device screen

5. Select the appropriate Ethernet card.
6. Click Forward.

The **Configure Network Settings** screen appears as shown in figure 8.4.

**Add new Device Type**

### Configure Network Settings

☒ Automatically obtain IP address settings with: **dhcp**

**DHCP Settings**

Hostname (optional):

☒ Automatically obtain DNS information from provider

☐ Statically set IP addresses:

**Manual IP Address Settings**

Address:

Subnet mask:

Default gateway address:

☐ Set MTU to:

**Cancel** **Back** **Forward**

Figure 8.4: Configure Network Settings screen

7. Select “Automatically obtain IP address with: dhcp” option.
8. Click Forward.

The **Create Ethernet Device** screen appears as shown in figure 8.5.



Figure 8.5: Create Ethernet Device screen

#### 8. Click Apply.

The Ethernet card, Intel Corporation 82801DB PRO/100 VE (LOM) Ethernet Controller is given a device name **eth0**. The IP address is obtained automatically with the dhcp (Dynamic Host Configuration Protocol).

### Exercise 2: Adding a host

#### Problem

Configure a host computer.

#### Analysis

The Domain Name Service (DNS) enables a computer to be located by its name. DNS resolves the host names provided by the users into their respective IP addresses when the users transact with a computer.

#### Solution

To configure a DNS client:

1. Select **System** → **Administration** → **Network** to open the **Network Configuration** screen.
2. Click the **DNS** tab.

The **DNS** tab of the **Network Configuration** tool appears as shown in figure 8.6.



Figure 8.6: DNS Tab

The hostname, domain, name servers and search domain are configured in the **DNS** tab.

### 3. Select the Hosts tab.

The **Hosts** tab appears, as shown in figure 8.7.

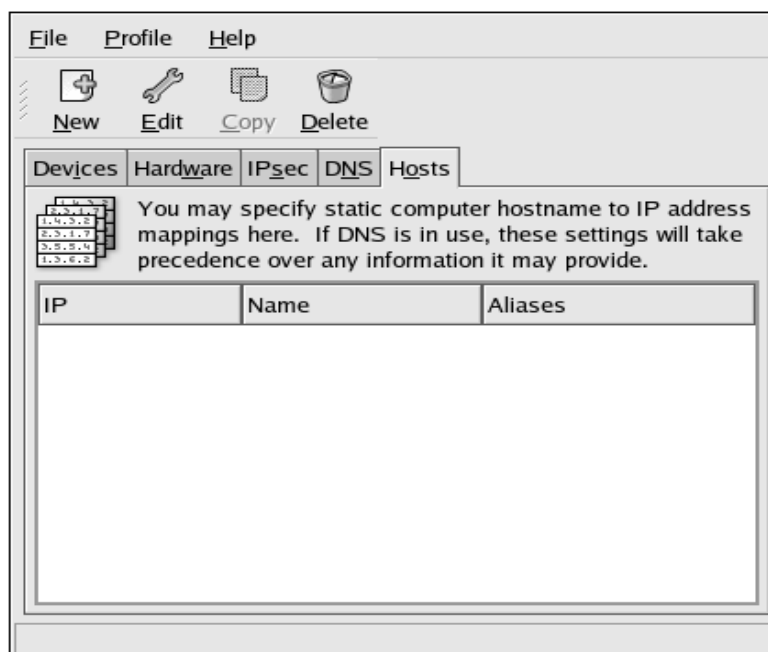


Figure 8.7: Hosts Tab

4. Click New to open the Add / Edit Hosts entry dialog box.
5. Specify the IP address, host name, and aliases for the host as shown in figure 8.8.

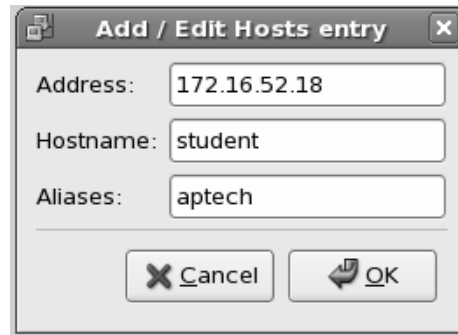


Figure 8.8: Add / Edit Hosts entry screen

6. Click OK to add a new host.

The student host is displayed in the **Hosts** tab as shown in figure 8.9.

7. Select File → Save.

The Information Dialog box appears.

8. Click OK.

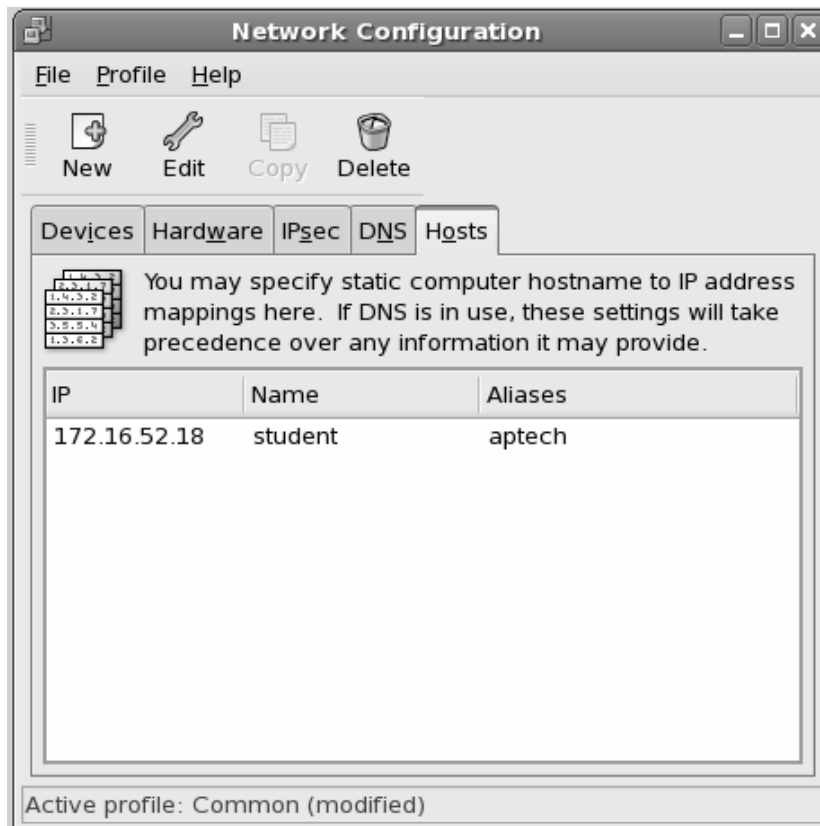


Figure 8.9: Hosts tab screen after adding a host

## Working with Red Hat Enterprise Linux 5.0

The Hosts tab allows you to add, edit, or remove hosts from the **/etc/hosts** file. This file contains IP addresses and their corresponding hostnames. When the system tries to resolve a hostname to an IP address, it refers to the **/etc/hosts** file before using the name servers. The **New** button in the Hosts tab is used to add an entry to **etc/hosts** file.

### Exercise 3: Configuring Samba Server

#### Problem

Configure the Samba Server and create user profiles to access the local system from the network.

#### Analysis

Samba Server enables you to create user profiles to access the local system from the network. Samba emulates a Linux system to appear as a Windows system for Windows to view files on a Linux system.

#### Solution

Consider an example in which you want to share the directory “examples” for the user “student” using Samba Server. The student user should have a Windows login access.

To create the user:

1. Select System → Administration → Server Settings → Samba.

The **Samba Server Configuration** dialog box appears as shown in figure 8.10.

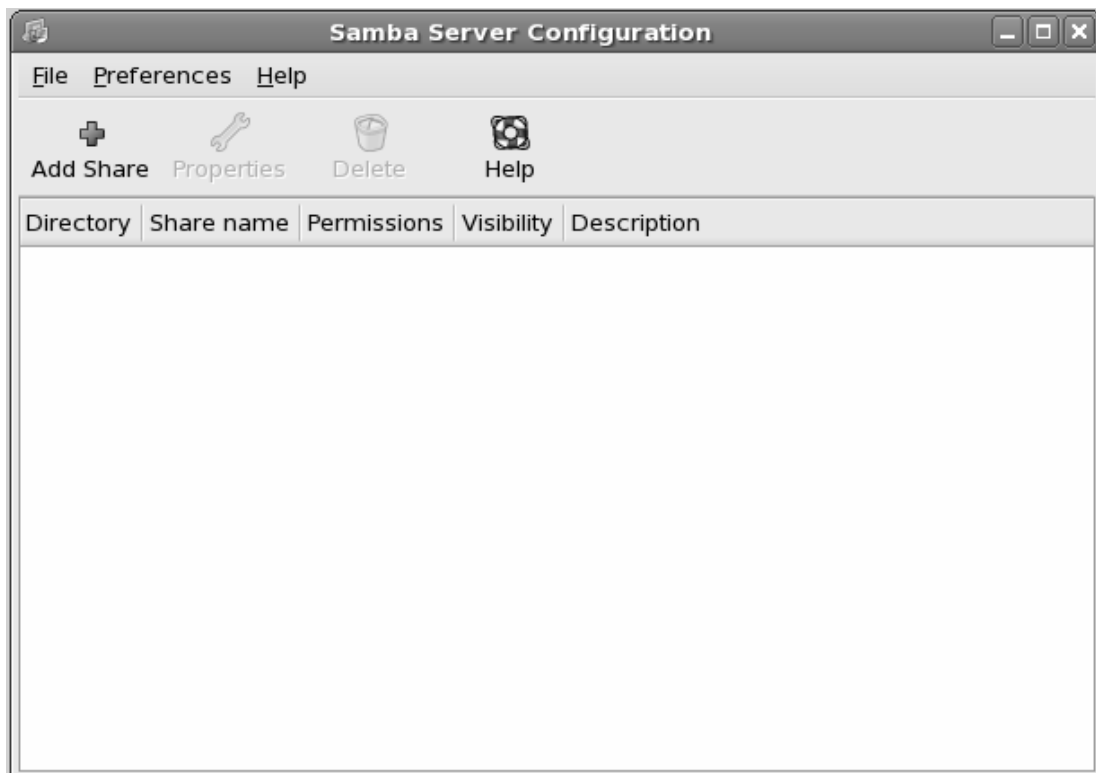


Figure 8.10: Samba Server Configuration dialog box

2. Select Preferences → Samba Users to add a new user for accessing the Samba share on Linux.

The **Samba Users** dialog box appears as shown in figure 8.11.



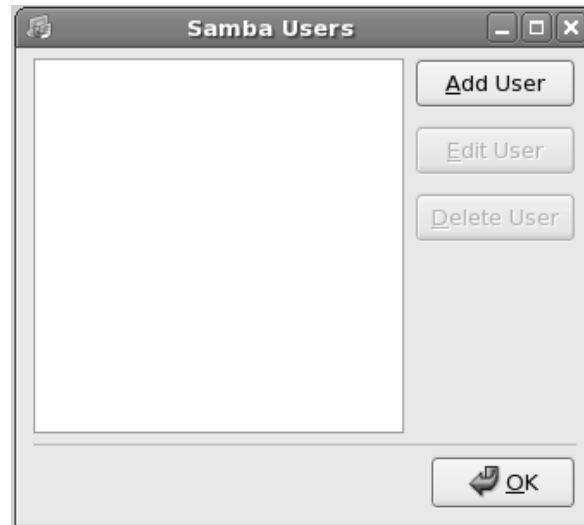


Figure 8.11: Samba Users dialog box

**3. Click Add User.**

The **Create New Samba User** dialog box appears as shown in figure 8.12.



Figure 8.12: Create New Samba User dialog box

4. Select the Unix Username as student from the list box.
5. Type the Windows username as student.
6. Type the password in Samba Password text box.
7. Confirm Samba Password text box.
8. Click OK.

The added User appears as shown in figure 8.13.

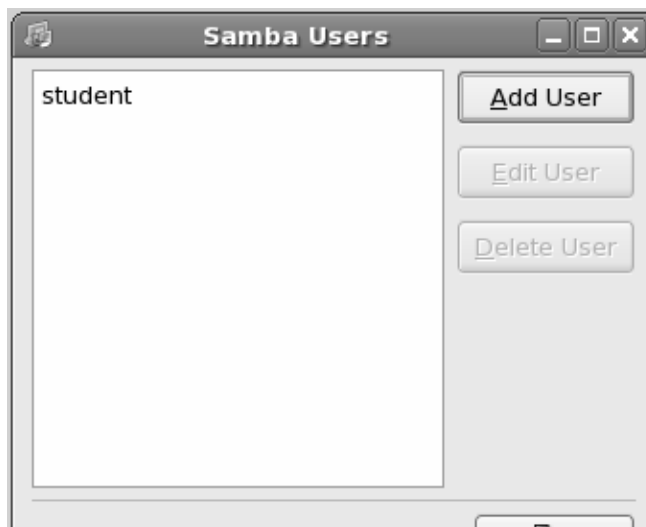


Figure 8.13: Added User

6. Click OK.

To configure the Samba Server:

1. Select **Preferences** → **Server Settings** in the Samba Server Configuration dialog box.

The **Server Settings** dialog box appears as shown in figure 8.14.



Figure 8.14: Server Settings dialog box

2. Type the workgroup name.

3. Click OK.

To share files on the Linux system:

1. Click the **Add Share** button in the Samba Server Configuration dialog box.

The **Create Samba Share** dialog box appears as shown in figure 8.15.



Figure 8.15: Create Samba Share dialog box

The **Writable** checkbox is checked if the user is given access to read and write to the shared directory. The **Visible** checkbox is checked if the user is to be granted read-only rights to the shared directory.

**2. Browse and select the directory that is to be shared.**

The Selected Directory appears as shown in figure 8.16.

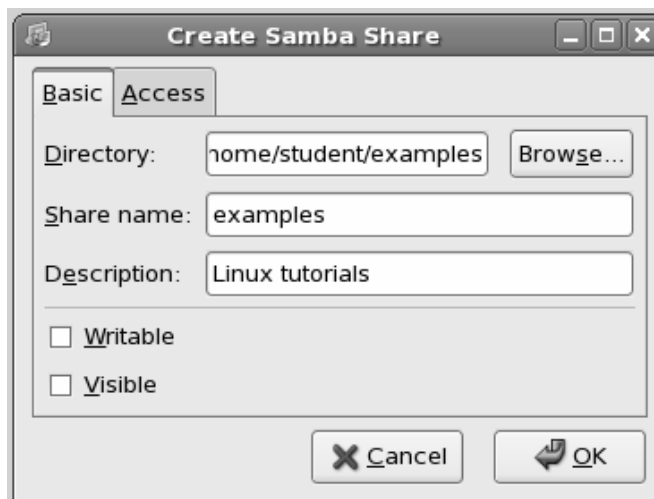


Figure 8.16: Selected directory

**3. Click the Access tab.**

The **Access** tab appears as shown in figure 8.17.

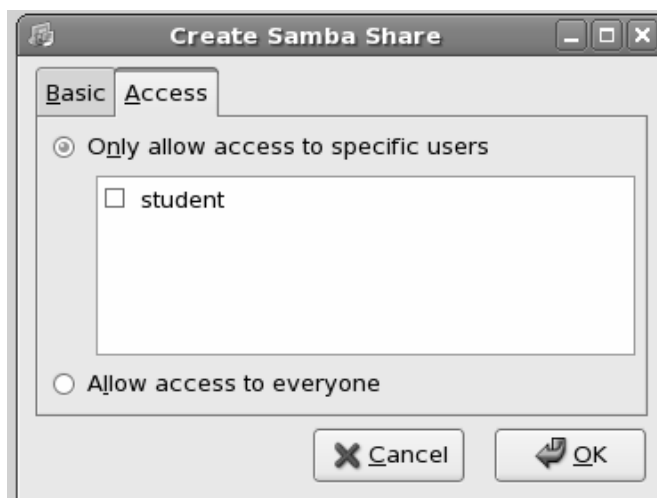


Figure 8.17: Access Tab

4. Select the user “student” to grant access to the shared directory.
5. Click OK.

The **Samba Server Configuration** dialog box appears as shown in figure 8.18.

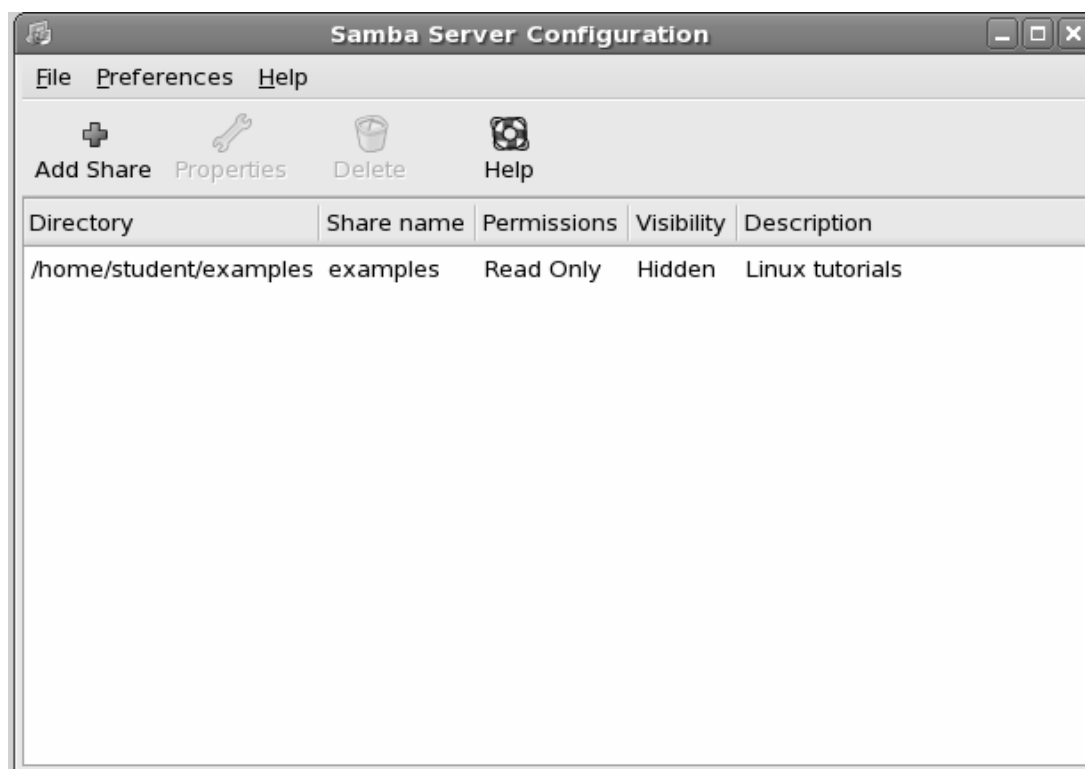


Figure 8.18: Shared directory added

The user ‘student’ is granted access to the shared folder ‘examples’. The visibility is ‘hidden’ thus giving read-only access to the user student.

**Exercise 4: Sending and Receiving mail using mail command****Problem**

Give the commands to send and receive mail.

**Analysis**

The `mail` command is used to send and read the e-mails. The `mail` command without any options reads all the mails in the current account. The `-f` option of the `mail` command reads a mail from a specific directory. The `-s` option specifies the subject on the command line.

**Solution****1. Login as student user. Use the mail command to list and display the mails in the student account.**

```
[student@localhost ~]$ mail
No mail for student
```

**2. Use the mail command with the option -f to read a mail in the specific directory.**

```
[student@localhost ~]$ mail -f /var/mail/student
Mail version 8.1 6/6/93. Type ? for help.
"/var/mail/student": 0 messages
&
```

**3. Use the mail command with the recipients e-mail address to send a mail in the specific recipient.**

```
[student@localhost ~]$ mail davidb@aptech.com
```

**4. Enter the Subject. Press <Enter>. Type the mail body and end the mail with a (.) dot.**

```
[student@localhost ~]$ mail davidb@aptech.com
Subject: Hi!
Hello, How are you?
.
Cc:
```

The (.) period at the end of the mail indicates that the email is complete and ready to be sent.

**5. Enter the name of the recipient in Cc.**

```
[student@localhost ~]$ mail davidb@aptech.com
Subject: Hi!
Hello, How are you?
.
Cc: martinp@aptech.com
```

**6. Use the mail command with the -s option to send contents of file (such as /tmp/message) as mail body.**

```
[student@localhost ~]$ mail -s 'Hi' davidb@aptech.com < /tmp/message
```

## Exercise 5: Viewing and Adding a Log File

### Problem

Write the steps to view and add a log file.

### Analysis

A system administrator should monitor the log files on a regular basis. Monitoring log files makes the administrator aware of different problems before they endanger the system. The system maintains the log files in `/var/log` directory.

### Solution

To monitor the system logs:

1. Login as root user. Select **System** → **Administration** → **System Log**.

The **System Log Viewer** screen appears as shown in figure 8.19.

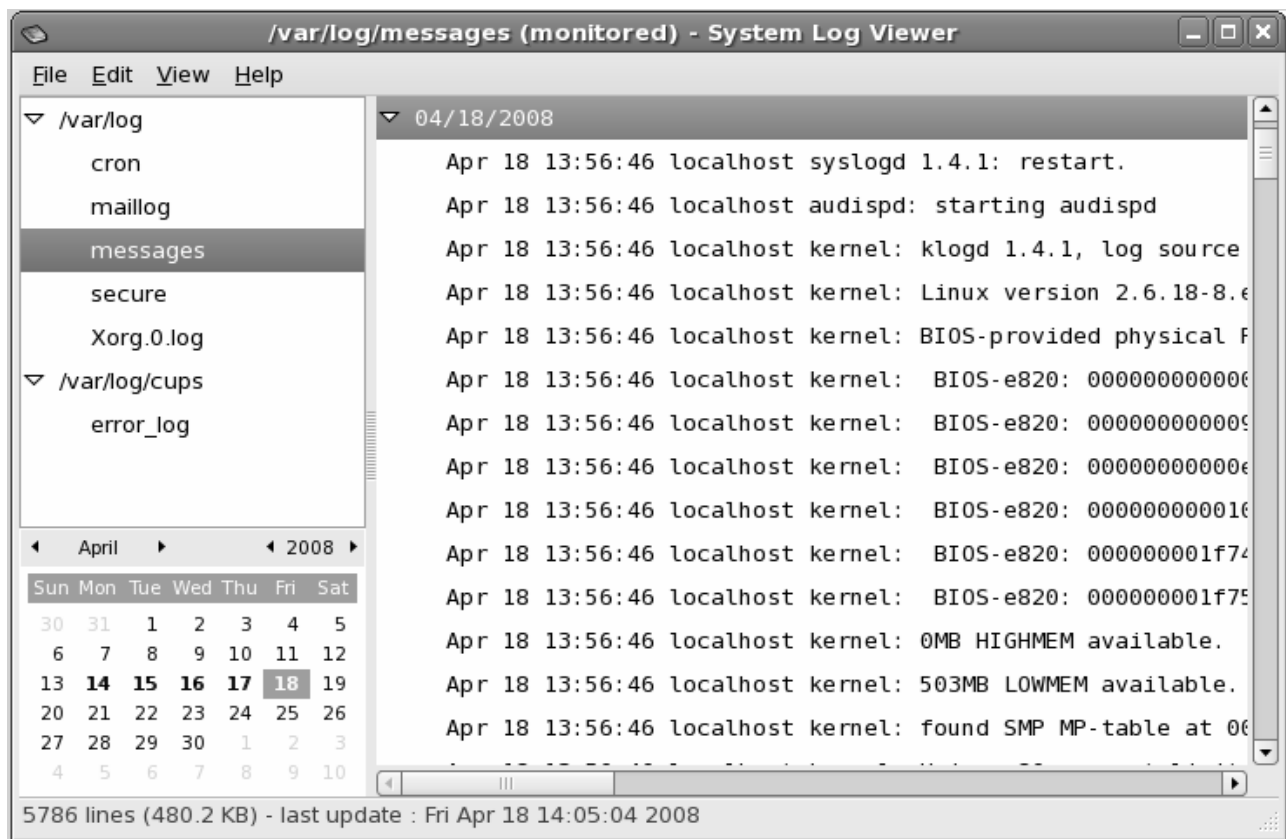


Figure 8.19: System Log Viewer screen

System logs are automatically created at the time of installation of various packages. The system logs enable verification of the installed packages, record any errors encountered during boot, and cron tasks. Cron is a utility that helps in automating tasks in Linux.

2. To add a log file, the user wants to view in the list, select **File** → **Open**.

This will display the **Open Log** window where the user can select the directory and filename of the log file that is to be viewed as shown in figure 8.20.

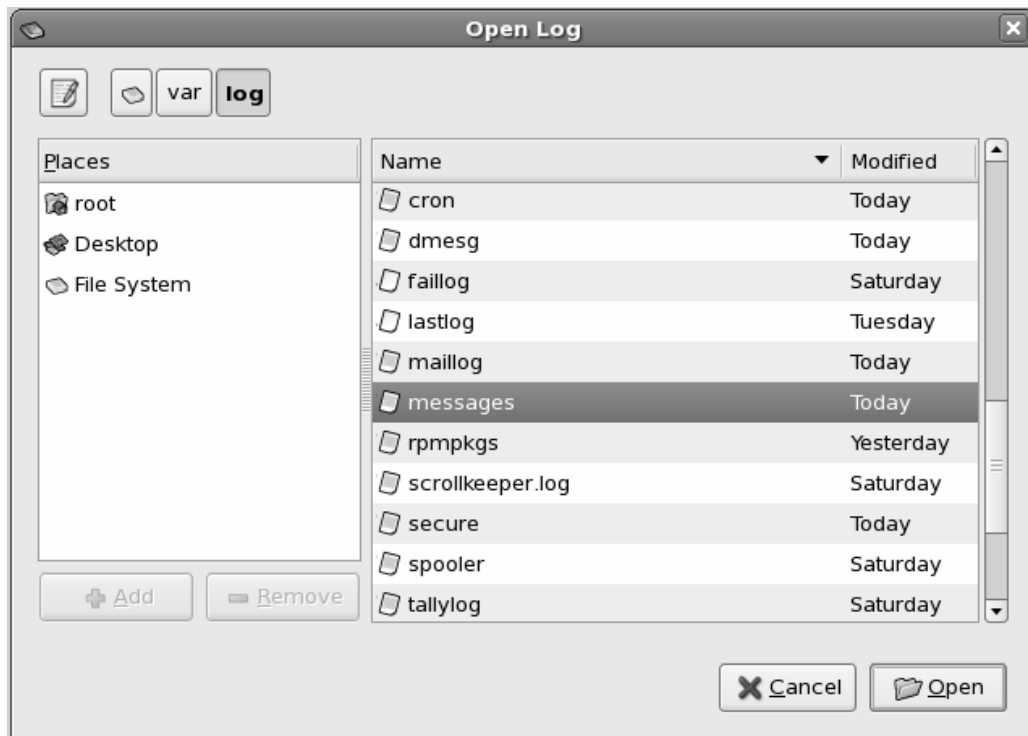


Figure 8.20: Adding a Log File

## Part II - 60 Minutes

1. Send an e-mail to your local account. Use `mail` and `mutt` utilities to read and send an e-mail.
2. Check the log files.
3. Venture and Co. has Linux as their operating system. You are a system administrator. Your employees want to access systems in the network using a domain name. Configure DNS for all of them.

## Do It Yourself

1. You have a user called Peter. Allow him to access a directory called Tutorials under /home/davidblake/. Configure Samba Server to achieve this task.
2. Send an e-mail with the following options:
  - a. Specify a subject line
  - b. Add a To: line to the message header for each recipient
  - c. Print the messages in first-in first-out order
3. Read an e-mail using `mutt` from /home/James.