

Essentials of Red Hat Linux

Are you registered with
Onlinevarsity.com?

Yes



No



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

Yes



No



Scores

- If Yes 50 marks each. If No 0 marks each.
- If you have **scored 100 marks**, you have a **legal copy of this ebook**.
- If you have **scored <100**, you have a **pirated copy of this ebook**.

Essentials of Red Hat Linux

© 2014 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

Edition 1 - 2014



Dear Learner,

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

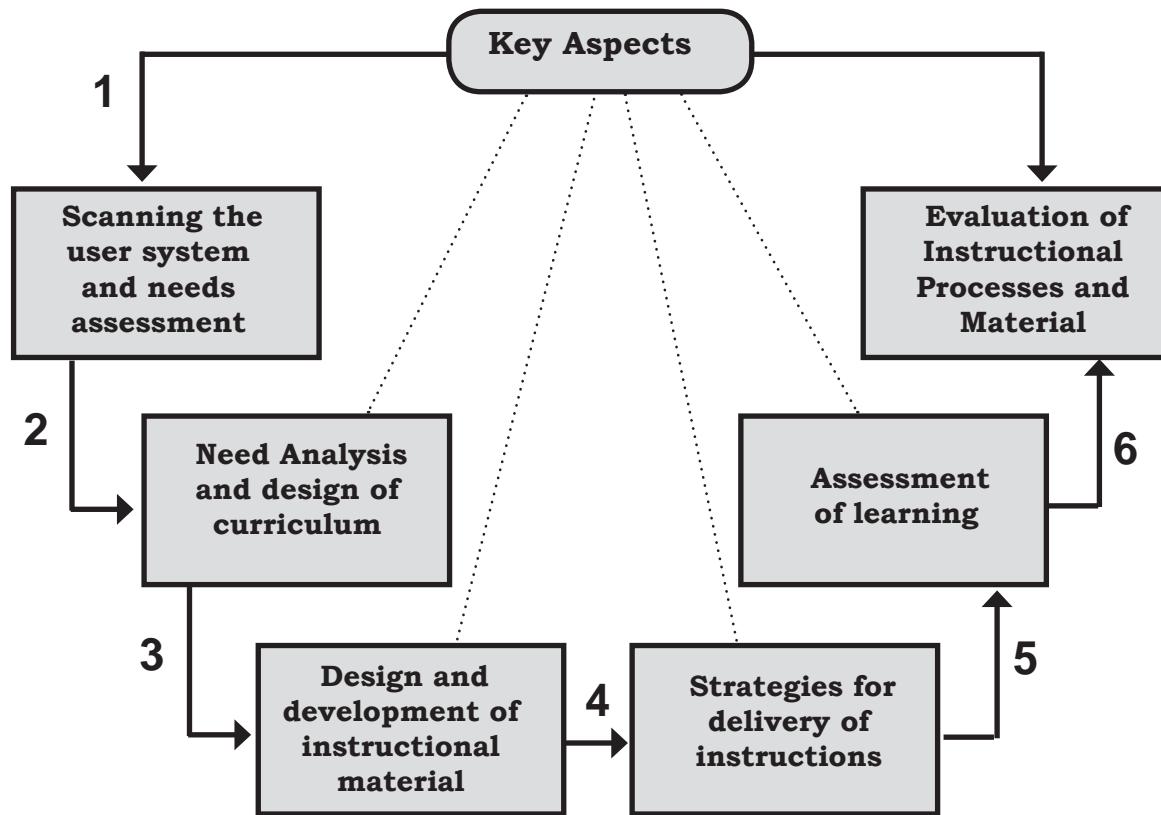
➤ Evaluation of instructional process and instructional materials

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



WRITE-UPS BY

EXPERTS AND LEARNERS

TO PROMOTE NEW AVENUES AND
ENHANCE THE LEARNING EXPERIENCE



FOR FURTHER READING, LOGIN TO

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 6.0. This book aims at making the students comfortable with the command line interface as well as graphical user interface. The students will also learn to work with and manipulate directories and files that includes, creating, renaming, copying, deleting, compressing and archiving directories and files. The course also enables the students to manage users, groups, printers and network connections. It also introduces the users with basic administration commands and shell scripting.

This book ‘Essentials of Red Hat Linux’ 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. Please send us your feedback, addressed to the Design Centre at Aptech’s corporate office.

We will be glad to receive your suggestions.

Design Team



Balanced Learner-Oriented Guide

for enriched learning available



www.onlinevarsity.com

Table of Contents

Sessions

1. Introduction to Linux
2. Linux Installation
3. Linux Installation (LAB)
4. Using Command-Line Interface
5. Using Command-Line Interface (LAB)
6. File System Basics
7. File System Basics (LAB)
8. Storage
9. Basic System Administration
10. Basic System Administration (LAB)
11. Shell Scripting
12. Shell Scripting (LAB)
13. Networking
14. Networking (LAB)
15. User Administration
16. User Administration (LAB)

ASK to LEARN

Questions
in your
mind?



are here to **HELP**

Post your queries in **ASK to LEARN** @

www.onlinevarsity.com

1.1 Introduction

An operating system is a software program that acts as an interface between a user and a computer. It consists of instructions that are given to the hardware components of the computer to perform specific tasks, such as creating or copying documents and viewing them. Some examples of operating systems are Linux, Windows XP, Windows 7, Solaris, Mac and UNIX.

Linux is primarily derived from the UNIX operating system. This operating system has evolved over a period of time with multiple features offering stability, high security and reliability. It runs on a variety of computer hardware, such as in mobile phones, tablets and mainframes. Due to these reasons, Linux has gained wide acceptance as a desktop operating system and a stable and secure server.

Another reason for Linux's popularity is that unlike proprietary operating systems, its source code is freely available to anyone. As a result, large communities of users and developers have contributed to make the operating system versatile in features. Many individuals and communities have used the Linux source code to develop different types of Linux-based operating systems, which are called distributions. Some of the popular Linux distributions are Debian, Fedora, Red Hat, SUSE and Ubuntu.

1.2 Defining Open Source

Open source software implies free software. The term free does not refer to the cost of the software. Instead, it refers to the freedom given to the users to suitably modify the software. The open source software allows users to freely use its source code and customize it to meet individual requirements.

Note - The open source software can be used as it is or can be redistributed after making modifications to it.

The open source software allows users to perform various tasks. Some of them are as follows:

- Access the source code of the software
- Run the software for commercial or personal use
- Customize the source code to meet specific requirements
- Redistribute copies of the software
- Release the customized software to the public in general

However, this redistribution of copies with or without modifications can or cannot be free. The person or agency redistributing the software can charge the end user the costs involved in media and redistribution.

The source code can also be improved and redistributed for a fee. Alternatively, the improved source code can be distributed free of cost but charge the users for services, such as upgrades, add-ons or support.

For example, Fedora Linux is available free of cost for end users along with the upgrades, enhancements

and add-ons to the operating system. However, Red Hat Enterprise Linux source code is free but end users have to pay a fee for upgrades, add-ons and support.

1.2.1 Comparing Open Source with Closed Source

Open source software is different from closed source or proprietary software. The proprietary software source code is not released to end users and is only available in a compiled executable state. The end users have to purchase the software to use it for personal or commercial purposes. In addition, copying or redistributing of the software can be restricted by the license agreement between the vendor and the end user.

Closed source software are generally owned, developed and distributed by a single vendor to end users who purchase the software for a license fee. Services, such as upgrades, add-ons and support are defined by the license agreement between the vendor and the end user. These can vary as per the selling and distribution strategy of the vendor.

1.2.2 Identifying Open Source Operating Systems and Applications

There are various open source operating systems and applications available today. Some examples of open source operating systems are as follows:

- **Open Solaris** - Is an open source operating system sponsored by Sun Microsystems.
- **Fedora Linux** - Is an open source Linux operating system sponsored by Red Hat. It has a release cycle of six months.
- **Red Hat Enterprise Linux (RHEL)** - Is an open source Linux operating system by Red Hat. However, end users have to pay subscription fee for support and other utilities shipped along with RHEL.
- **Open SuSE** - Is an open source Linux-based operating system sponsored by Novell. Open SuSE is a German distribution. There are a lot of applications and utilities packed along with Open SuSE Linux. It also contains a customized edition of StarOffice and Word Perfect. It is very popular in Europe.
- **Mandrake Linux** - Is an open source Linux operating system. Mandrake offers the power and stability of Linux to both individual and enterprise users in a user-friendly environment. It contains various high-quality applications, such as a complete Microsoft Office suite of programs. It also provides a built-in installation support.
- **Debian Linux** - Is an open source Linux-based operating system sponsored by the Debian community. Debian is among the earliest distributions of Linux and is available free of cost. It offers precompiled software bundled with more than 8000 packages and has an easy installation procedure.

The various open source applications are as follows:

- **Apache Web Server** - Is an open source Web server sponsored by Apache Software Foundation. It runs on various operating system platforms including UNIX, Linux, Solaris and Windows. It also provides support for a variety of features ranging from server-side programming language to multiple authentication schemes.
- **Sendmail** - Is an open source mailing application sponsored by the Sendmail consortium. It is a general purpose Internetwork e-mail routing application. It also supports various types of mail transfer and delivery methods, such as Simple Mail Transfer Protocol (SMTP). Sendmail is also available in many proprietary versions.
- **MySQL** - Is an open source database application sponsored by Sun Microsystems.

1.2.3 Licensing

An open source license is a copyright license that allows the source code of software to be available for everyone to use and modify. End users of the software can troubleshoot and customize the source code as per their own requirements. There are many types of licenses that legalize the use of open source software for various purposes.

Some examples of open source licenses are as follows:

- **General Public License (GPL)** - Allows users to package and redistribute the original or modified source code of the software only with other GPL Licensed software. It does not allow the open source software to use any other proprietary software.
- **Lesser General Public License (LGPL)** - Allows users to package and redistribute the original or modified source code of the software only with other LGPL Licensed software. In addition, it allows the use of proprietary software with LGPL Licensed open source software.
- **Berkeley Software Distribution (BSD)** - Allows redistribution of the open source software with any other software. It allows distribution of original source code after modification even if it uses some other software not covered under the BSD license. In addition, it allows the use of proprietary software with BSD covered software.

1.3 Origin of Linux

Earlier, developers were looking for an operating system that was simple, elegant, flexible and secure. At that time, the developers of Bell Laboratories introduced an operating system that fulfilled all these requirements. This gradually evolved into the UNIX operating system. Linux was derived from UNIX to provide a free or very low-cost operating system to computer users. It is considered to be efficient and fast-performing operating system.

1.3.1 History of Linux

The name of the Linux operating system was coined by Linus Torvalds in 1991. Linux was developed on Minimal UNIX (MINIX), a UNIX version, using GNU Not UNIX's (GNU's) C compiler.

UNIX

In 1965, Bell Labs and Massachusetts Institute of Technology (MIT) along with General Electric developed one of the first multiuser computer systems, called Multiplexed Information and Computing Service (Multics). Ken Thompson and Dennis Ritchie, two Bell Labs engineers, worked on this project till 1969.

Later, they developed another operating system in C language. This added flexibility in porting the operating system to other computing platforms, which was not easy with Multics as it was written in assembly language. This operating system was named Uniplexed Information and Computing Service (Unics), which was similar to the Multics operating system. The name Unics was then changed to UNIX.

In 1972, Ken Thompson and Dennis Ritchie re-wrote UNIX using the C programming language. Around 1974, UNIX was licensed to universities for educational purposes and, a few years later, it was made available commercially.

Berkeley Software Distribution (BSD)

During 1976–1977, Ken Thompson was on sabbatical for six-months from Bell Labs. He taught UNIX operating system as a visiting professor at the Computer Science Department at the University of California-Berkeley (UCB). His sessions were very popular.

After Thompson's return to Bell Labs, students and professors at Berkeley continued to enhance UNIX and finally these enhancements were incorporated into BSD Version 4.2.

MINIX

MINIX was another version of UNIX that was used as a teaching aid in universities and colleges. It was a copy of the UNIX operating system available with the source code. Due to its small size, micro kernel-based design and ample documentation, it was well suited to people who wanted to run a UNIX-like system on their personal computer.

MS-DOS was created much later than UNIX. By that time, the industry had begun to accept UNIX as the standard operating system. Therefore, UNIX features have influenced the design of MS-DOS. Many vendors, such as Sun, IBM and Hewlett-Packard purchased the source code of UNIX. They developed their own versions of UNIX.

This resulted in many differences amongst various versions of UNIX offered by various organizations. To avoid the confusion, some standards called Portable Operating System Interface (POSIX), were outlined. POSIX is a set of standards that enabled software to run on various UNIX-based operating systems without changing the source code.

Note - POSIX was produced by the Institute for Electrical and Electronic Engineers (IEEE) and is recognized by International Organization for Standardization (ISO) and American National Standards Institutes (ANSI).

In 1983, an MIT scientist, Richard M. Stallman, launched the GNU project. The main aim of this project was to create a UNIX-like operating system but free from licensing charges. The next step of this project was to further improve the operating system. For this, the GNU operating system was distributed to programmers around the world. The GNU operating system used GRand Unified Bootloader (GRUB) for booting.

For a better coordination of the GNU project, Stallman and other people created the Free Software Foundation (FSF). This organization promoted the development and use of free software. FSF developed the GPL for building free software protected from those entities that would use it to create proprietary closed-source systems. The organization charges a small fee to operate the foundation. Around 1990, FSF had developed a number of tools that could be freely modified and redistributed.

1.3.2 Evolution of Linux

Around 1991, one of the students from the University of Helsinki, Finland was working on MINIX. His name was Linus Torvalds. He wanted to modify his own version of the UNIX operating system by using its existing features. After he created this operating system, he gave it free of charge for everybody to use. He named his project as Linux and wrote the source code for the kernel, which was the core program of the Linux operating system. After writing the source code, he made the Linux kernel available on the Internet.

The kernel was then combined with the GNU system and as a result, a new operating system was discovered, the GNU/Linux operating system. This operating system is also referred to as the Linux operating system. Like other operating systems, it has a kernel in its core. The kernel controls the resources of the computer and forms an interface between the user and the hardware.

The source code of the Linux kernel was made available to everyone on the Internet for study and modification purpose. Whenever any modifications were done to the kernel code, Linus accepted them. As a result, whenever a new version of Linux with new functionality is released, anyone can use it and can also help in fixing bugs, if any. To maintain stability, Linus ensured that the new code is quality tested. The new code is then merged into the kernel. This entire cycle is a part of the development model.

However, in the closed model, a project team works on a software version, test it and fix the errors before everyone else use the software. By looking at the version number of Linux, it can be known if it is a stable or trial version. For example, in version 1.a.b, if 'a' is an even number, it confirms that this is a stable version of Linux. Alternatively, if 'a' is an odd number, it indicates that this is a beta (trial) version of Linux.

The official mascot for Linux is the Linux penguin, called Tux. This mascot was selected by Linus Torvalds.

Figure 1.1 shows the Linux official mascot.



Figure 1.1: Linux Official Mascot

1.4 Basic Principles of Linux

Like, UNIX, the Linux operating system also works on the certain principles. These basic principles of the Linux software are as follows:

- Stability and Reliability
- Security

Stability and Reliability

Linux is a stable operating system, which does not require to be restarted periodically to maintain performance levels. It provides adequate protection against freezing up or slowing down over time due to issues, such as disk defragmentation and clogging of temporary files. This results in increased uptime.

Linux is modular by design. It can only install applications that are really required. This saves a lot of resources that would go waste in managing unnecessary applications and utilities that otherwise come bundled with non-modular operating systems. It also helps Linux to remain highly customizable and easy to maintain and manage.

As a multiuser operating system, everything in Linux is managed as a file and all the configurations are stored in text format. The software components of Linux are single purpose programs and are designed to work together to perform complex tasks in a multiuser environment. This helps Linux perform better even with less resources.

The various utilities and programs of the Linux operating system are majorly open source. A large community of developers and users are there to test, modify and improve them. Therefore, bugs are reported and fixed relatively faster as compared to proprietary operating systems. A large developer

base also results in frequent upgrades and improved releases to take care of any shortcomings in the operating system.

Security

Linux provides a number of features that cater to various security requirements at personal, commercial and organizational levels.

The various Linux security features are as follows:

- **File System Security** - Linux has a hierarchical file system that keeps similar files, such as data, programs and configuration files at fixed locations in a structured way. This allows better placement of security policies to ensure safety for the file system.
- **Firewall Security** - Linux provides a highly customizable and secure firewall that can cater to various personal and commercial requirements. It supports very simple yet effective configurations to very complex configurations catering to commercial or organization scenarios.
- **Run Levels Security** - Linux provides run levels with a predefined set of services being available in particular run levels. These run levels can also be customized to include or exclude services suiting to user requirements. The run levels help users initiate and run minimum number of services required. This not only helps in saving precious computing resources but also reduces surface area for any security threats.
- **Services** - Linux services are known as daemons. These daemons can be configured to run under fine grained user level security. For example, users in Linux can only run the services meant to be run by them. This prevents the possibility of an attack that can compromise the entire computer at a time.
- **Security Enhanced Linux** - Linux provides Security Enhanced Linux (SELinux) that provides better access control and security at the kernel level through the security policies. SELinux runs applications into various sandboxes and isolates them from each other and the underlying operating system.
- **Discretionary Access Control (DAC)** - Linux uses DAC to ensure security in a multiuser environment. The files and directories are labelled with a set of permissions to indicate the user or group that the object belongs to and the types of operation they can perform on it, such as read, write or run.

Advantages of Linux Operating System

The Linux operating system has several advantages. Some of the advantages are as follows:

- **Reliability** - Linux servers need not be shut down for years together. This implies that users do not face any operating system failures.

- **Backward Compatibility** - Linux supports older hardware. It can run on various types of processors, such as 386 and 486 Intel processors. In addition, it can run on DEC Alpha, Sun SPARC, PowerPC and SGI MIPS.
- **Simple Installation and Upgrade Process** - Most Linux versions are menu-driven and have easy installation procedures. In addition, it provides the ability to upgrade from prior versions easily. In the upgrade process, the system uses the existing configuration files and maintains a list of its actions during installation.
- **Low Cost of Ownership** - The total cost of implementing Linux server software is low. In addition, the cost of support can also reduce as there are a lot of people and organizations providing free support for Linux. The system configuration requirements for installing a Linux computer are less. For this reason, the hardware and maintenance cost goes down.
- **Support for Legacy Devices** - Linux can also run on a computer that has low configuration.
- **Graphic User Interface (GUI)** - X Window System is the graphical interface for Linux. It is divided into two subsystems consisting of a server and a client. The various GUIs provided by Linux are K Desktop Environment (KDE) and GNU Network Object Model Environment (GNOME). Both of these GUIs are the versions of the X Window System.
- **Excellent Security Features** - Linux offers a secure firewall, access control and customized run levels to ensure security.
- **Support for Development Libraries** - Linux offers various platforms for many development languages, such as C, C++, JAVA, Python and Perl. It also supports Integrated Development Environments (IDE), such as KDevelop and Glade.
- **Support for High User Load** - Linux can support a large number of users working simultaneously. The limitation is typically because of the server hardware.

1.5 Features of Linux

The Linux operating system provides various features, such as multitasking, shared libraries, POSIX compliance and virtual memory. Some of these features are as follows:

- **Multiprogramming** - Allows programs to be executed simultaneously using the same set of computing resource. In Linux, multiprogramming is made possible by using the concept of time-sharing. The operating system has to manage various programs to be executed. These programs run in a queue and CPU time is shared among them. The active program gets the CPU time for a specific period. After that is again lined up in the queue to wait for its turn again. In the meantime, the CPU attends the next program in the queue.

- **Multitasking** - Is the ability of any operating system to handle the execution of multiple tasks. When a task is waiting for the completion of an activity, the CPU begins with the execution of the next task. When a task is waiting for the user to provide inputs, another task can be uninstalling any program/application. An example of multitasking could be a scenario where a person can have breakfast, watch TV and talk to someone on the phone simultaneously. Here, the person are performing more than one task simultaneously. However, at a given point in time, the person would be taking a bite, watching the TV or speaking on the phone. The time among all the tasks was divided. Similarly, the CPU divides the time among all the active tasks. The kernel is responsible for scheduling the tasks.
- **Virtual Memory** - Is an additional memory that an application can use when the physical memory is already consumed. Some part of the hard disk is made available for the application, to be used as the virtual memory. This is because to run large applications or run multiple applications at a time, the amount of physical memory is not always be sufficient. Therefore, the system places infrequently required programs and data in the virtual memory. The system loads the programs in the memory, whenever required.
- **Shared Libraries** - Are sets of functions or subroutines that are maintained as a set of files. With the help of this feature, multiple applications are not required to individually maintain their code for any function. They can access the required functions from the shared library files. This saves hard disk space and memory.
- **POSIX Compliance** - Supports most of the standards set for all UNIX systems.
- **Samba** - Is derived from the Server Message Block (SMB) protocol of the Microsoft operating systems. This feature is used to share file and print services. With the help of Samba, the user can share a Linux file system with the Windows operating system. Alternatively it also helps users to share the Windows file system with Linux operating system.

Note - A protocol is a set of rules that states the communication standards for data transfer between two applications.

The term file system refers to a mechanism used by an operating system to store and manage files and directories on a storage medium.

A partition is a logical portion of the hard disk, which is created using either the operating system-based utility or any third-party utility.

- **Network Information Service (NIS)** - Allows the creation of user accounts that can be shared across multiple computers in a network. It is a client and server database system that acts as a central database of account information used for account authentication.
- **Office Suites** - Are supported by Linux. Basically, Linux supports OpenOffice.org, which is a set of programs that has many built-in tools. OpenOffice.org enables a user to create documents,

presentations and illustrations and also analyze data. OpenOffice.org provides a set of applications just like Microsoft Office Application for Windows.

- **Data Archiving Utilities** - Are the utilities provided by Linux for basic data backup (archiving), such as tar, cpio and dd. Advanced Maryland Automatic Network Disk Archiver (AMANDA) is a backup system supported by Linux. It enables the LAN administrator in setting up a master backup server and make backup for multiple hosts in a large capacity tape drive.
- **Web Server** - Linux comes with the Apache Web server, one of the most popular Web servers in use today. In addition, Apache supports the Squid proxy server. It helps in improving the performance for accessing the Internet.
- **Licensing** - Licensing implies that anyone can download, install or use the software and any updates to the software, regardless of the delivery mechanism. Linux is a copyright under the GNU GPL.
- **Additional Features** - Include useful and free software, such as text editors, browsers and scientific applications.

1.6 Red Hat Operating System Distributions

Red Hat Inc. provides different variants for various types of users. It has divided its users into the following two categories and they are as follows:

- **Business Users** - The first category includes the business users requiring an operating system that requires fewer updates. Red Hat provides the Red Hat Enterprise Linux product suite for these users. These users require to pay a license fee for the software. The changes in the operating system are made at a low pace so that business users are not required to upgrade the operating system at frequent intervals.
- **Other Users** - The second category includes the users who use Linux for experimental purpose and make use of its new and advanced features. These users use the software without paying any license fee. They can download the software from the Internet and use it free of charge. For example, Fedora is a freely downloadable operating system.

The various Linux variants provided by Red Hat Inc. are as follows:

- **Red Hat Enterprise Linux Suite** - Runs on multiple hardware architecture such as IBM PowerPC, AMD64, Intel EM64T and Intel x86/x64. The operating systems of the Red Hat Enterprise Linux suite are as follows:
 - **Red Hat Enterprise Linux AS** - Used for large servers and supports up to 16 CPUs. This operating system supports databases, Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) applications.

- **Red Hat Enterprise Linux ES** - Used for small and middle range servers and supports up to two CPUs. It is used for business applications, such as printing, mailing and networking applications.
 - **Red Hat Enterprise Linux WS** - Used as desktop operating system and supports up to two CPUs. This operating system is compatible with Red Hat Enterprise Linux AS and Red Hat Enterprise Linux ES. This operating system is suitable for client applications, such as document processing and software development applications.
- **Red Hat Desktop** - Is used for small and medium business groups, supporting a single CPU. It is compatible with Red Hat Enterprise Linux AS, Red Hat Enterprise Linux ES and Red Hat Enterprise Linux WS. This operating system is suitable for document processing, Web browsing, software development and instant messaging.
- **Fedora Core** - Represents a general-purpose operating system with capabilities comparable to other operating systems, such as Windows NT and Macintosh. Fedora Core is a Red Hat-sponsored project but Red Hat Inc. does not support it. An individual can download Fedora free of charge from various Web sites. The official Web site for Red Hat is <http://www.redhat.com> Red Hat Enterprise Linux versions released so far are as follows:
- ➔ Red Hat Enterprise Linux 2.1 (Panama)
 - ➔ Red Hat Enterprise Linux 3 (Taroon)
 - ➔ Red Hat Enterprise Linux 4 (Nahant)
 - ➔ Red Hat Enterprise Linux 5 (Tikanga)
 - ➔ Red Hat Enterprise Linux 6 (Santiago)

Red Hat Enterprise Linux is closely related to the Fedora Linux community project sponsored by Red Hat. Each Red Hat Enterprise Linux version takes technologies from Fedora after considerable stabilization and quality assurance effort. While Fedora core has a six months release cycle, RHEL has a more conservative release cycle of two years.

Even earlier, RHEL versions were also based on Red Hat's open source Red Hat Linux operating system. But afterwards, Red Hat Linux split into community-based Fedora project and commercial Red Hat Enterprise Linux. The latest RHEL 6 has many features of Fedora 13 and Fedora 14.

However, Fedora is a community project that serves as an upstream version for RHEL. RHEL serves as a commercial enterprise operating system that is released as alpha and beta versions before final release.

As RHEL is an open source operating system and its source code can be accessed by anyone, there are certain operating systems that use RHEL source code. Users compile their own versions of RHEL. One such popular operating system is CentOS. These rebuilds provide free updates from non-Red Hat servers.

However, these operating systems do not get any support from Red Hat.

1.7 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 6.0 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 6.0 are:

- ❖ **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 6.0 are:

- ❖ **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 open motif 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.

1.8 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:
```

1.9 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.

1.9.1 Root User

The root user is a special administrative account which 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.

1.9.2 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.

1.9.3 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 1.2.

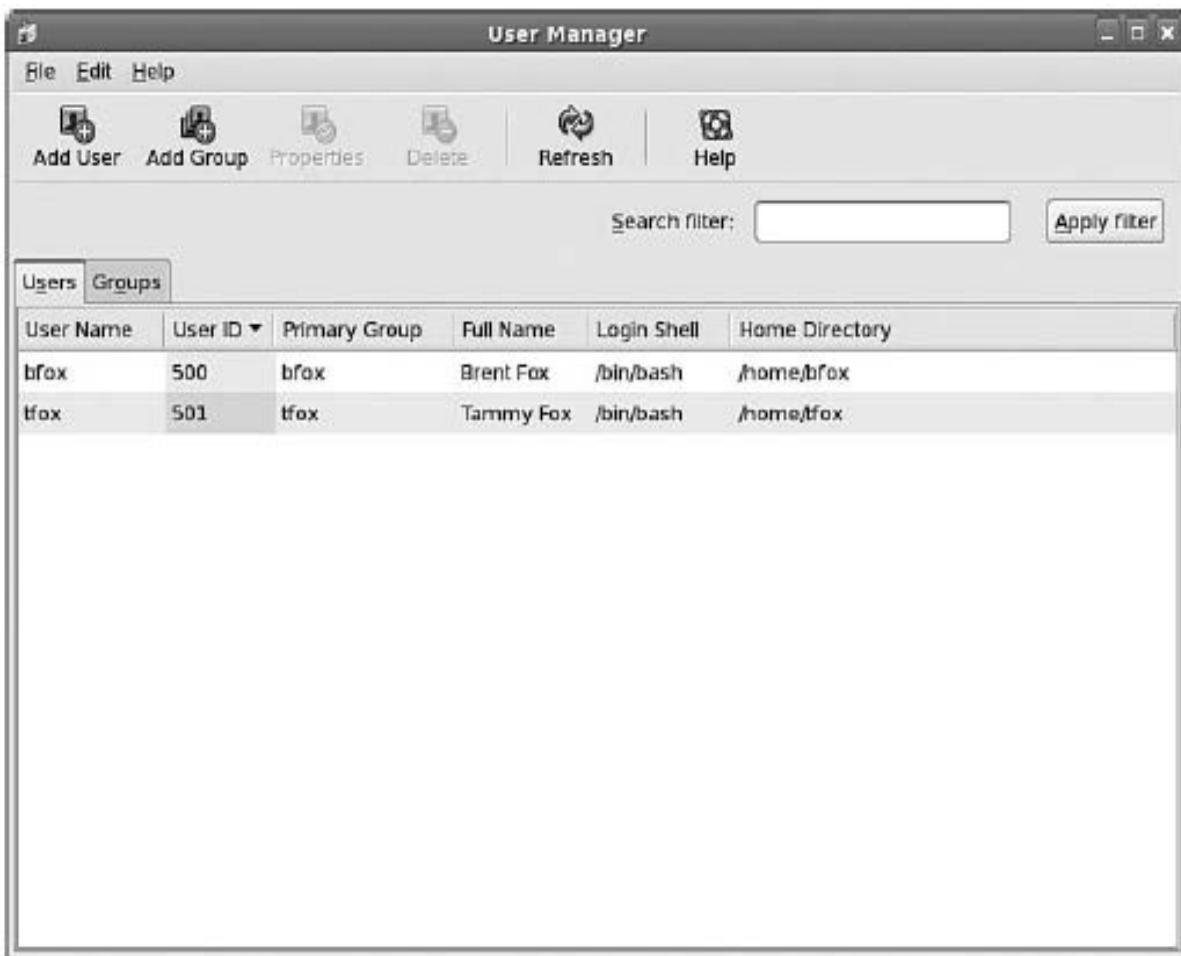


Figure 1.2: List of Existing Users

A new user can be added to the system by clicking the add user button in the user manager window. Create new user window is displayed as shown in Figure 1.3. 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.

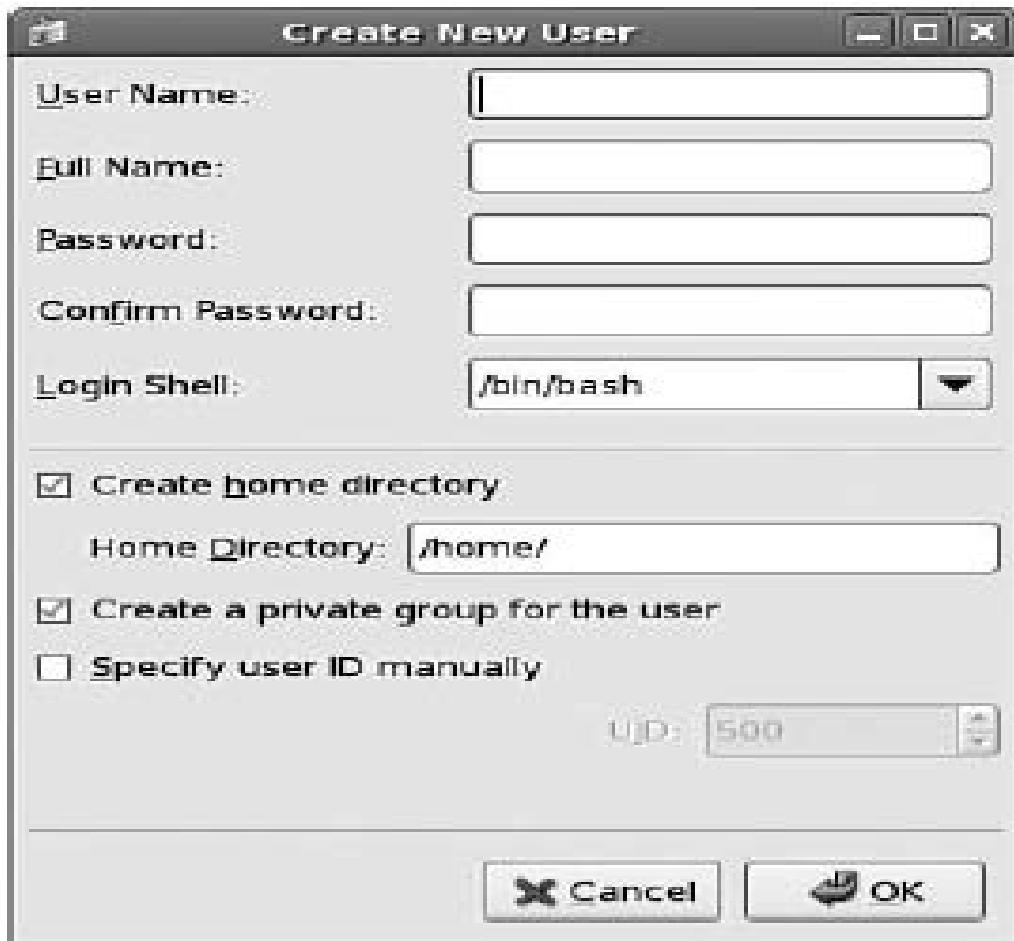


Figure 1.3: Adding a New User

1.9.4 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 1.4.

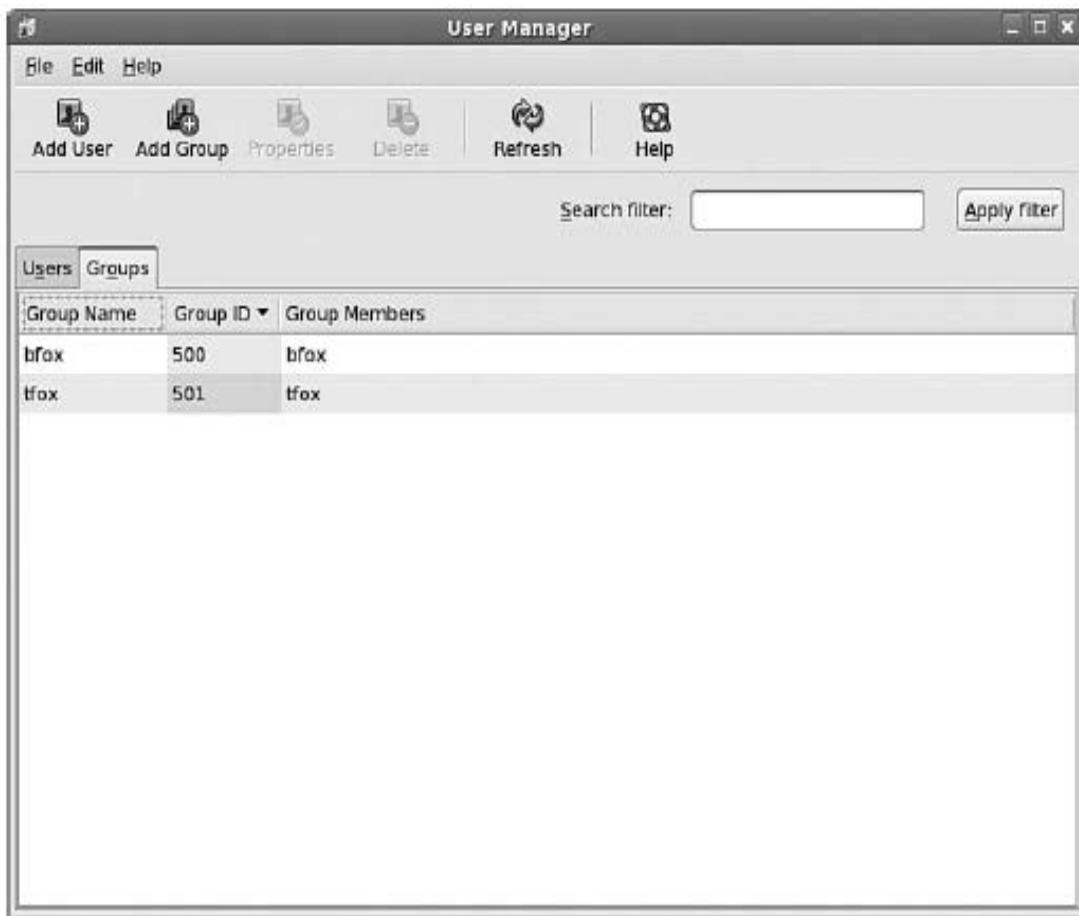


Figure 1.4: List of Existing Groups

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 1.5. 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 checkbox is checked, the user can select a group ID from the list box.



Figure 1.5: Adding a Group

1.10 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.

1.10.1 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.

1.10.2 **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
```

1.10.3 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
```

1.10.4 Reading Directory and Files

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

→ **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:

- ✧ **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#	Function
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
```

→ `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.d
rwxrwxr-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.d
rwxrwxr-x 5 bball bball 1024 Nov 5 10:03 research
-rwxrwxr-x 1 bball bball 200 Oct 24 13:24 search
```

→ 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 ~]$
```

→ **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.

Options	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.

→ 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 $
```

→ 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
```

→ 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.

→ 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	Reboot after shutdown
-c	Cancels a pending shutdown

Table 1.11: Options of the `shutdown` Command

1.11 Advanced Linux Commands

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

1.11.1 **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 file systems.

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
```

1.11.2 **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 S TIME      TTY      TIME     CMD
student   2840     2809      0 16:16 pts/0 00:00:00 ps -f -a
```

1.11.3 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:

- ➔ 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

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**.

1.11.4 **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.localdomain 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.

1.11.5 `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.

1.11.6 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:

→ **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

Options	Function
-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 user name, 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 -Pkill 385
```

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

1.11.7 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 demonstrates the same.

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.

1.12 Check Your Progress

1. Which of the following operating systems is derived from UNIX?

(A)	Linux	(C)	MAC Operating system
(B)	Windows XP Professional	(D)	Novell Netware

2. Open source software is _____.

(A)	Proprietary software	(C)	Closed software
(B)	Paid software	(D)	Free software

3. Who sponsored the open source operating system, Open Solaris?

(A)	Microsoft	(C)	MAC
(B)	Sun Microsystems	(D)	Novell Netware

4. What is the name of the official mascot for Linux?

(A)	Tux	(C)	Geeko
(B)	GNU	(D)	Beastie

5. Which of the following features of RHEL allow programs to be executed simultaneously using the same set of computing resource?

(A)	Multi-booting	(C)	Multitasking
(B)	Multiprogramming	(D)	All of these

6. To display all the files, including the hidden files in the current directory, the _____ option of the ls command must be used.

(A)	-A	(C)	-l
(B)	-a	(D)	-F

7. To display the date in the **mm/dd/yy** format, the _____ option of the date command must be used

(A)	%m	(C)	%D
(B)	%d	(D)	%y

8. When a new user is added to the system, a _____ is automatically formed.

(A)	public user group	(C)	duplicate user group
(B)	private user group	(D)	secure user group

9. The _____ command is used to display the currently running processes in the Linux system.

(A)	public user group	(C)	show
(B)	private user group	(D)	ps

10. The _____ command is used to combine two or more files.

(A)	cat	(C)	ls
(B)	combine	(D)	more

1.12.1 Answers

1.	A
2.	D
3.	B
4.	A
5.	C
6.	C
7.	C
8.	B
9.	D
10.	A



Summary

- An operating system is a software program that acts as an interface between a user and a computer. Linux is an operating system, which is derived from the UNIX operating system. It is an open source operating system. Open source software is free while closed source software is proprietary.
- Linux was developed on Minimal UNIX (MINIX), a UNIX version, using GNU Not UNIX's (GNU's) C compiler. The official mascot for Linux is the Linux penguin, called Tux. Some of the features of Linux operating system include multiprogramming, multitasking, virtual memory and shared libraries. It also provides various advantages such as reliability, stability and low cost of ownership.
- 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: date, who, man, dir, vdir, tree, cat, and more commands for reading directories and files command, exit command and shutdown command.
- The advanced Linux commands include:
 - df, ps, kill, uname, tty, printing commands such as 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.

TechnoWise



Are you a
TECHIE GEEK
looking for updates?

Logon to

www.onlinevarsity.com

2.1 Installing Linux

The following scenario helps users to understand the Linux installation process.

XYZ Inc. is a leading software development organization based in Atlanta. Steve works as a system administrator with the organization at their recently opened office at Denver. The branch office at Denver is working on a software development project with a local client.

The development team is assisted by the support teams that include IT, marketing and operations located at the Denver office. Each support team consists of five employees.

Steve, who is part of the development team, is asked to set up a file server where developers can save their work. In addition, this server also works as a test staging server for different applications being developed by the developers.

Therefore, Steve decides to set up **RHEL** server to work as file and application hosting server for the purpose. Steve's expectations for the server are as follows:

- The server to be robust enough to take higher workloads
- The server to be modular by design
- The server that is less prone to virus attacks as compared to other operating systems
- The server that is highly reliable and scalable for hosting enterprise applications

To set up the server, Steve should plan the installation tasks and identify the suitable pre-installation requirements. These requirements are as follows:

- Required version of RHEL
- Compatibility of the hardware with the selected distribution of Linux

2.1.1 Identifying the Pre-requisites for Installation

Before installing RHEL, Steve has to ensure that the computer meets the minimum hardware requirements. These include a compatible processor, bus, memory, modem, printer, Network Interface Card (NIC), hard disk, video card, mouse, keyboard and CD-ROM drive. Ensure that devices, such as display and NICs, used during installation are compatible with the distribution of installed Linux.

These devices are as follows:

- **Processor** - Linux supports various Intel CPUs, such as Pentium II, Pentium III, Pentium IV and non-Intel processors, such as AMD, Cyrix, Alpha AXP, PowerPC and SPARC. Linux also supports Symmetric Multiprocessing (SMP) that enables the use of more than one processor on a computer. The user can enable the SMP option while compiling the Linux kernel. Linux provides support for multiple CPUs on one computer.

- **Bus** - Linux supports the Peripheral Component Interconnect (PCI), VESA local (VL-bus) and Micro Channel Architecture (MCA) buses. It also supports hot swappable buses. These buses can be attached or removed from system without turning OFF the computer. Some examples of hot swappable buses are the USB, FireWire and Personal Computer Memory Card International Association (PCMCIA) peripheral buses for laptops.
- **Memory** - Minimum 512 MB of RAM is required to install RHEL 6.0.
- **Modem** - Linux supports a wide range of internal and external serial modems.
- **Printer** - Linux provides support for a majority of serial and parallel printers.
- **Small Computer System Interface (SCSI) Controller** - Linux supports various types of SCSI that are used for connecting SCSI peripheral devices to a computer. The user should install a SCSI controller card to connect the SCSI devices to a computer. A SCSI controller card can be plugged into a connector slot available on a bus.
- **Hard Drive** - Linux supports the IDE hard drives that the BIOS of a computer support without using any driver. The user can also connect a SCSI hard drive to a computer through a SCSI controller card. A driver that can enable a SCSI controller to access a SCSI drive should be installed on the Linux operating system.
- **NIC** - Linux supports the Ethernet network cards, ARCnet and IBM token ring network. When the user is required to install Linux on an existing network, ensure that a network card is installed on the computer to connect to the network. The network card should be compatible with the Linux version that is installed on the computer.
- **Video Card** - Linux supports a wide range of video cards, such as, Super VGA and IBM monochrome. When the X Window System is used, the user should use a video card supported by XFree86. The support of XFree86 for a video card depends on the video chipset that is used to control the monitor display.

2.1.2 Choosing the Type of Installation

After understanding the hardware and software requirements for RHEL, the user determines whether to perform an upgrade or a new installation.

For example, consider that a user is instructed to install RHEL 6.0 on a server that already has RHEL 5 installed on it. In this case, the user can always choose to upgrade the server. However, when performing the upgrade, the user determines the upgrade path from one version to another version.

Upgrade is typically recommended when users are moving to the next version. It becomes difficult to upgrade from a very old version to the new version because of incompatibilities that can exist between the old version and the new version.

When users upgrade their computer, the system installs updated versions of the packages. The upgrade process preserves the existing configuration files by renaming them with an **.rpmsave** extension.

Note - It is always advisable to backup the required data. For example, if the OS is undergoing up-gradation process, the technician should backup the required data on another hard drive. The considerations for upgrading the computer with RHEL 6.0 are as follows:

- There can be a problem with individual package configuration files. Due to the changes in the configuration file formats and layouts, the individual package configuration file can or cannot work.
- There can be some applications or products installed on the computer, which are required to be upgraded manually.
- There can be problems with the third-party applications installed on the computer.
- There can be a possibility of the system not restarting after the upgrade. In this case, the recovery of the data can become difficult.

2.1.3 Choosing the Mode of Installation

When users start the computer to install RHEL, they must choose an appropriate installation method. The selected method depends on how users access the installation files.

Table 2.1 lists the supported installation methods.

Installation Method	Description
Using a CD-ROM/ DVD	This method uses a CD-ROM/DVD that stores the installation files for RHEL. By default, the installation program searches for the IDE CD-ROM/DVD drive to initiate the installation. However, if users are using the SCSI CD-ROM/DVD drive, they must select a SCSI driver during the installation. The CD-ROM method should be used if users have to install RHEL on a single computer or users do not have a setup ready for installing it over a network.
Using a hard drive	This method uses the RHEL installation files stored in a folder on the hard drive of a local computer for Linux installation. The installation files are stored in the same folder structure of a CD-ROM/DVD in the form of an ISO image. This enables the installation program to access this directory to install Linux. The hard drive method requires that the hard disk used for installation should have an operating system installed on it. This method of installation is the quickest because the installation files are locally stored on the hard disk. The hard drive method is used only if users do not have the startup CD-ROM to install Linux.

Installation Method	Description
Using an Network File System (NFS) server	This method uses an NFS server that stores the Linux installation files. The NFS server exports the directory that contains Linux installation files to the system when users install Linux. Alternatively, the NFS server can export the mirror image of the Linux installation files. Users should configure the NFS server to access the installation files to install Linux. Users can use the NFS method if they have to install Linux over LAN. NFS method is useful when users have a large number of installations to perform. For example, Steve has to install Linux on all the computers on the network. Steve should use the NFS method if NFS is already configured on the network.
Using an File Transfer Protocol (FTP) server	This method uses an FTP server to install Linux. For the installation, users should access the installation files stored in a directory on the FTP server. FTP method should be used when there are a large number of installations to be done.
Using an Hyper Text Transfer Protocol (HTTP) server	This method uses the HTTP (Web) server to install Linux. For the installation, users should access the installation files stored in a directory on the HTTP server. Users can access the HTTP server by using its name or IP address. HTTP method should be used when there are a large number of installations to be done.

Table 2.1: Linux Installation Methods

Serial Console Installation

The serial console installation method is commonly used on servers and computers without an interface, such as a monitor or a keyboard. For example, Sun SPARCs are installed using this method.

To enable a serial console installation, two computers are required, one on which the Linux installation takes place and the other from where the installation is initiated and controlled. A null modem cable between the serial ports is used to connect both these computers. A terminal script is executed on the computer controlling the installation, to provide access to the console of the server.

The following command should be executed to initiate the installation:

```
boot: console=<device>
```

In the given command, the <device> parameter specifies the device name of the serial port, such as `ttyS0`.

2.1.4 Troubleshooting Installation Issues

There can be situations where users can face issues while installing RHEL. In such situations, users are required to troubleshoot those issues to proceed with the installation. Table 2.2 lists some of the common troubleshooting problems along with the appropriate solutions.

Problem	Description/Symptom	Solution
Unable to boot RHEL	Users can face a situation where they just performed installation but cannot restart the computer.	Reinstall and rearrange disk partitioning.
Computer displays signal 11 error	<p>This type of error is also referred to as segmentation fault. This can occur due to reasons that are as follows:</p> <ul style="list-style-type: none"> ❖ One of the programs tried to access the memory location that was not assigned to it. ❖ There is a bug in one of the software programs installed on the computer. ❖ There is a hardware error in the memory of the system's bus. 	<p>The steps to be preformed are as follows:</p> <ol style="list-style-type: none"> 1. Ensure that users have the latest installation updates and images from Red Hat. 2. If this solution does not work, it can be a hardware issue. These issues are generally related to the computer's memory or CPU- cache. 3. A possible solution for this error is to turn off the CPU-cache in the BIOS. 4. Swap the computer's memory in another motherboard slot to check if the problem is with the slot or memory. 5. Check the installation media.
Facing issues with booting into the graphical installation	There can be issues with some video cards that can lead to problems with booting into the graphical installation. In such a scenario, if the installation software does not run using its default settings, it tries to run in a lower resolution mode. Alternatively, the installation software attempts to run in text mode.	<p>Possible solutions are as follows:</p> <ol style="list-style-type: none"> 1. To adjust the resolution from the boot option. 2. To update the video card drivers. <p>There can be a possibility that the correct video card drivers were not loaded with the operating system boot up. Installing correct video card drivers can solve this problem.</p>
No devices found to install RHEL Error Message	If this error occurs, there can be a problem that the SCSI/IDE controller is not being recognised by the installation program.	One of the possible solutions is to check the hardware vendor's Web site to determine if the drivers are available.

Problem	Description/Symptom	Solution
Using all the available free space	Users have created a swap and a / root partition and selected the root partition to use the remaining space, but they still see unused hard disk space.	In case the hard drive is more than 1 GB, users can use all of the remaining space on the hard drive by creating a boot partition.

Table 2.2: Linux Troubleshooting Problems with Solutions

2.2 Logging on to the Red Hat Enterprise Linux System

After starting the Linux system, a login prompt that appears is as follows:

```
Red Hat Enterprise Linux Server release 6.0 (Santiago)
Kernel 2.6.32-71.e16 on an i686
<server name> login: _
```

In the prompt, **<server name>** indicates the **hostname** of the system. The host name is also known as the login name or user name. Users must enter the user name at the login prompt. After entering the user name at the login prompt, users must enter the password.

```
Red Hat Enterprise Linux Server release 6.0 (Santiago)
Kernel 2.6.32-71.e16 on an i686
<server name> login: root
Password:
```

Linux stores all the user names and user information in special files namely, shadow and password. The login name and password are checked against these files. Only authorized users are provided access to the computer. If any unauthorized user enters a login name that does not match the user names stored in the file, the login message is displayed again.

Login name and password are case-sensitive and have to be entered carefully. The entire login process appears as follows:

```
Red Hat Enterprise Linux Server release 6.0 (Santiago)
Kernel 2.6.32-71.e16 on an i686
<user name> login: root
Password:
Last login: Mon Oct 21 11:18:01 from 171.15.58.123
[user name]> root]$ _
```

In the code, **[user_name@linuxpc1current_directory_name]\$** is the symbol displayed at

the shell prompt, by default when a valid user name is entered. Here, **user_name** is the user's login name and **current_directory_name** is the user's current working directory.

2.2.1 Working with the GNOME Terminal

GNOME is a user-friendly desktop environment that can run on multiple operating systems. It is the default desktop environment for RHEL. GNOME is an open source desktop environment and it consists of a set of X clients built to support an X desktop environment.

2.2.2 Working with GNOME

The default window manager for GNOME is Metacity. In addition, GNOME can run with other window managers that are GNOME-compliant, such as Enlightenment and Sawfish. Being GNOME-compliant indicates being aware of the GNOME features, such as unit management, the GNOME Pager and getting support for multiple desktops. Users can also run non-compliant window managers on the GNOME desktop environment with a negligible loss of functionality. The GNOME desktop is divided into three distinct areas and they are as follows:

- ➔ **Top Menu Panel** - The grey bar at the top of the desktop.
- ➔ **Desktop Area** - The working area in the centre that fills most of the screen.
- ➔ **Bottom Menu Panel** - The grey bar at the bottom. It is also known as the Windows List Panel.

Figure 2.1 displays the GNOME desktop.



Figure 2.1: GNOME Desktop

Top Menu Panel

The left side of the top menu panel contains the following default components:

- **Applications Menu** - This menu contains a hierarchy of submenus, from which applications that are installed on the system can be started. These applications can be updated as and when software is installed or removed. When a user clicks on the Applications menu, the submenu is displayed.
- **Places Menu** - This menu provides the user places or locations to store the personal files, folders or documents.
- **System Menu** - This menu provides user access to configuration tools, access to help documentation and allows the user to log out or shut down and many more. Figure 2.2 shows the System menu.



Figure 2.2: System Menu

- **Application Launchers** - These are series of icons located next to the menus. These icons help to access commonly used applications. These icons are as follows:
- **Mozilla Firefox** - This is the first icon from the left side of the top menu panel and is a Web browser.
 - **Evolution** - This is the second icon from the left side of the top menu panel. It is a mail client and personal information manager. The Evolution icon is shown in Figure 2.3.

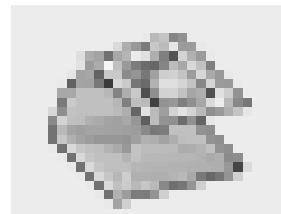


Figure 2.3: Evolution Icon

- **Gnote** - This is the third icon from the left side of the top menu panel. This icon helps in taking notes. The **Gnote** icon is shown in Figure 2.4.

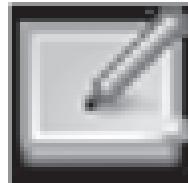


Figure 2.4: Gnote Icon

To add more launchers to a panel, a user should right-click the panel and select Add to Panel. A user can also add launchers for the applications that are in the Applications menu. To do this, right-click the application that has to be added and then select Add this launcher to panel.

The right side of the top menu panel contains the default components that are as follows:

- **Notification Area** - The notification area displays icons that alert a user about the new events that have taken place on a computer. For example, an e-mail program displays an icon to inform the user about the arrival of a new e-mail. Alternatively, a printer program displays an icon to show the user that a document is being printed.
- **Volume Control and Sound Preferences Applet** - The volume control applet allows a user to change the overall volume of sound on the computer. Users should click the volume control icon. It displays a slider and with the help of mouse, the slider is dragged to the desired volume. When users double-click the icon, a volume control window appears that allows user to separately change the volume of different sound sources.
- **Network Manager Applet** - The network manager applet allows a user to quickly connect and disconnect from the available networks, both wired and wireless. Users can see what connections are enabled or available when they click the applet. It also allows users to edit or remove saved connections when they right-click the applet.
- **User Switch Area** - The user switch area shows the current username, located next to the clock. With a click, it provides options to a user to enter information about their accounts. It also shows system preference option that takes the user to the Control Center window and options to lock the screen and quit from the system.
- **Clock and Calendar Applet** - A user can see the calendar when they click the clock on the right side of the panel. When they double-click a date, it opens the Evolution personal information manager. It also has a weather feature. In addition, it allows users to modify or add locations that should be displayed.

Desktop Area

- The default desktop area or workspace contains three icons. They are as follows:
- **Computer Icon** - This icon lists the available storage devices on a computer. Users must double-click this icon to open the window that lists the devices. To access these devices click, Places Computer menu.
- **User Home Icon** - When users double-click the icon, a window appears that contains all the logged-in user's files, such as music, movies and documents, which are stored by default. Each user has a separate home directory and users cannot access each other's home directories by default.
- **Trash Icon** - This icon refers to the Trash folder from which users can access a file that has been deleted. If users have to remove a file permanently, they can empty the trash by selecting the Empty Trash option.

Bottom Menu Panel or Window List Panel

The bottom menu panel has three components and they are as follows:

- **The Window List Panel** - The window list panel displays any open application whether visible or hidden. The applications are displayed as small buttons. For a hidden window, the button shows a white background while the currently selected button shows a grey background. To switch between windows, users can click the application's button in the panel.
- **The Workspace Switcher** - The workspace switcher is located on the right side of the bottom menu panel. The switcher allows a user to move between work spaces. There is a separate desktop for each workspace and a corresponding window list panel. Users can organize applications in the workspace.
- **The Trash Icon** - The trash icon is located on the right side of the window list panel. When users delete a file, it is moved to this area. Users can use options such as Open the folder, Empty Trash, Right-click the Trash icon and remove from Panel and Lock to Panel.

2.3 Adding Software Packages

Users have to install a software package that was not installed during the installation. This can happen due to requirements arising after the operating system has been installed.

In such situation, users can install software packages manually using the rpm command.

Software packages for Red Hat supported or sponsored operating systems are made in Red Hat Package

Manager (RPM) format. The way software packages for Windows operating system are made available in .exe or .msi format, Red Hat software packages are made available in .rpm format. These software packages are also called rpms or rpm packages.

Finding RPM Packages

RHEL provides the **rpm** command to manage the RPM software packages. Users can use the **rpm** command to install, update, remove or query information about rpm software packages.

Users can find a number of rpm packages for different utilities in the rpm repositories. The rpm repositories are as follows:

Red Hat Enterprise Linux CD/DVD - Is a repository of all the rpm packages shipped with the Red Hat Enterprise operating system.

Red Hat Errata Page - Is a repository of rpm packages shipped with the different versions and variants of operating systems by Red Hat. It is available at <http://www.redhat.com/apps/support/errata/>.

Red Hat Network - Is an online repository of rpm packages for the different versions and variants of operating systems by Red Hat. It is an Internet-based single point of management solution RHEL systems. It is available at an annual subscription fee.

Using the **rpm** Command to Install Packages

Users can use the rpm command to install rpm packages from the command line. The following command can be used to install an rpm package:

```
rpm -ivh<complete path of the rpm package>
```

where,

-i option indicates that the command involves installation of the rpm package in question.

-v option indicates that the execution of the command is verbose and messages are printed on the screen.

-h option indicates that # marks are printed to show the progress of any operations being performed by the command.

The following example illustrates the installation of an rpm package demopackage-1.0-1.i386.rpm:

```
#rpm -ivh demopackage-1.0-1.i386.rpm
```

Output:

Preparing..... ##### [100%]

1: demopackage-1.0-1.i386.rpm ##### [100%]

Using the **rpm** Command to Update Packages

Users can use the following command to update an rpm package:

```
rpm -Uvh<complete path of the rpm package>
```

where,

-U option indicates that the command involves updating the rpm package in question.

-v option indicates that the execution of the command is verbose and messages are printed on the screen.

-h option indicates that # marks are printed to show the progress of any operations being performed by the command.

Using the **rpm** Command to Query Installed Packages

Users can use the following command to query all the installed packages:

```
rpm -qa
```

where,

-q option indicates that the command involves querying an rpm package

-a option indicates that the command involves querying all the installed rpm packages.

To check if a particular installation package is installed users can use the following command:

```
rpm -qa | grep <name of the rpm package>
```

Using the **yum** Command to Install RPM Packages

Users can use the following command to install an rpm package with yum:

```
yum install <complete path of the rpm package>
```

To search an rpm, users can execute the following command:

```
yum search <name of the rpm package>
```

2.4 Check Your Progress

1. Which of the following installation methods use the Linux ISO images for installing Linux?

(A)	NFS	(C)	CD/DVD
(B)	Hard drive	(D)	FTP

2. Which of the following component interact directly with the application on screen without the need for a mouse or keyboard?

(A)	64 MB	(C)	512 MB
(B)	128 MB	(D)	100 MB

3. Which of the following components of the GNOME desktop environment provides user access to configuration tools, access to help documentation and allows user to log out or shut down?

(A)	System menu	(C)	Applications menu
(B)	Places menu	(D)	Application launchers

4. Which of the following components of the GNOME desktop environment allows user to store personal files, folders or documents?

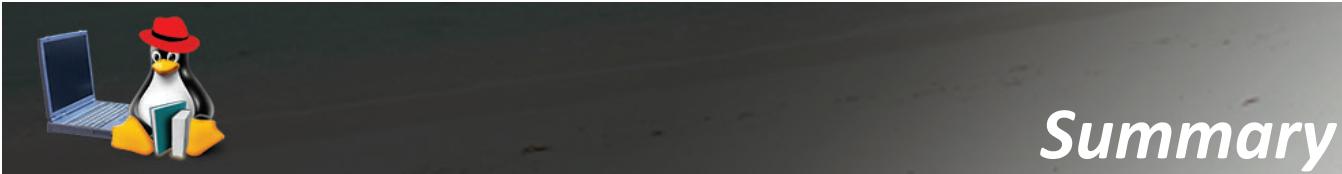
(A)	System menu	(C)	Applications menu
(B)	Places menu	(D)	Application launchers

5. Which of the following `rpm` command is used to update packages?

(A)	<code>rpm -Uvh <complete path of the rpm package</code>	(C)	<code>rpm -qa</code>
(B)	<code>rpm -ivh <complete path of the rpm package></code>	(D)	<code>yum install <complete path of the rpm package></code>

2.4.1 Answers

1.	B
2.	C
3.	A
4.	B
5.	A



- Before installing Linux, ensure that the computer meets the minimum hardware requirements. These include a compatible processor, bus, memory, modem, printer, Network Interface Card (NIC), hard disk, video card, mouse, keyboard and CD-ROM drive.
- Users can either choose to upgrade their computer or do a fresh installation. When users upgrade their computer, the system installs updated versions of the packages, which are currently installed on their computer. There are different methods to install Linux operating system, such as, DVD, hard drive, NFS, FTP or HTTP.
- In addition, users learned how to work on a GNOME terminal. GNOME is a user-friendly environment that can be run on multiple operating systems. The GNOME desktop consists of various components, such as Top Menu panel, Bottom Menu panel and desktop icons. The Top Menu panel consists of a number of components, such as Applications Menu, Places Menu, System Menu, Date and Time display and so on. The Bottom Menu Panel also consists of a number of components, such as Workspace Switcher and Trash Icon.
- Lastly, users learned that they can install software packages manually using the rpm command. RHEL provides the rpm command to manage the rpm software packages.

Are you looking for online **HELP?**

We are just a *click* away



To chat with a

login to **www.onlinevarsity.com**

Exercise - 1**Installing Red Hat Enterprise Linux 6.0**

Typical installation of RHEL takes around 1.5 hours. If the setup seems to hang, wait at least 10 minutes before restarting the system. Installation can experience momentary screen blackouts. To ensure successful installation, ensure that the hardware is fully compatible with Red Hat Enterprise Linux 6.0. Users can verify the list of supported hardware from <http://hardware.redhat.com/hcl/>.

Before proceeding with the installation, users can modify the BIOS settings to allow them to boot from the CD-ROM, which is the preferred method of installation for this lab. In this lab, users install RHEL as Desktop.

Step 1 - Turn ON the computer. Insert a licensed copy of RHEL and reboot the system. Users are then prompted to install Red Hat Enterprise Linux (Refer to Figure 3.1).



Figure 3.1: Red Hat Enterprise Linux 6 Installation Screen

Step 2 - Press ENTER to start the setup. RHEL boots the system with the boot process (Refer to Figure 3.2).

```

Freeing SMP alternatives: 15k freed
ACPI: Core revision 20090903
Enabling APIC mode: Flat. Using 1 I/O APICs
..TIMER: vector=0x30 apic1=0 pin1=2 apic2=-1 pin2=-1
CPU0: Intel(R) Core(TM)2 Duo CPU      T5470  @ 1.60GHz stepping 0d
Brought up 1 CPUs
Total of 1 processors activated (3202.00 BogoMIPS).
devtmpfs: initialized
regulator: core version 0.5
NET: Registered protocol family 16
ACPI: bus type pci registered
PCI: MCFG configuration 0: base e0000000 segment 0 buses 0 - 255
PCI: MCFG area at e0000000 reserved in E820
PCI: Using MMCONFIG for extended config space
PCI: Using configuration type 1 for base access
bio: create slab <bio-0> at 0
ACPI: BIOS _OSI(Linux) query ignored
ACPI: Interpreter enabled
ACPI: (supports S0 S1 S4 S5)
ACPI: Using IOAPIC for interrupt routing
ACPI: No dock devices found.
ACPI: PCI Root Bridge [PCI0] (0000:00)
pci 0000:00:07.3: quirk: region 1000-103f claimed by PIIX4 ACPI
pci 0000:00:07.3: quirk: region 1040-104f claimed by PIIX4 SMB

```

Figure 3.2: RHEL Boot Process

Step 3 - On the next screen, click OK. This starts media test as shown in Figure 3.3. It usually takes a few minutes to perform the media test. Alternatively, the disc media test can also be skipped. The setup process initiates anaconda, which is the Red Hat Enterprise Server Installer (Refer to Figure 3.4).



Figure 3.3: Starting Media Test Screen

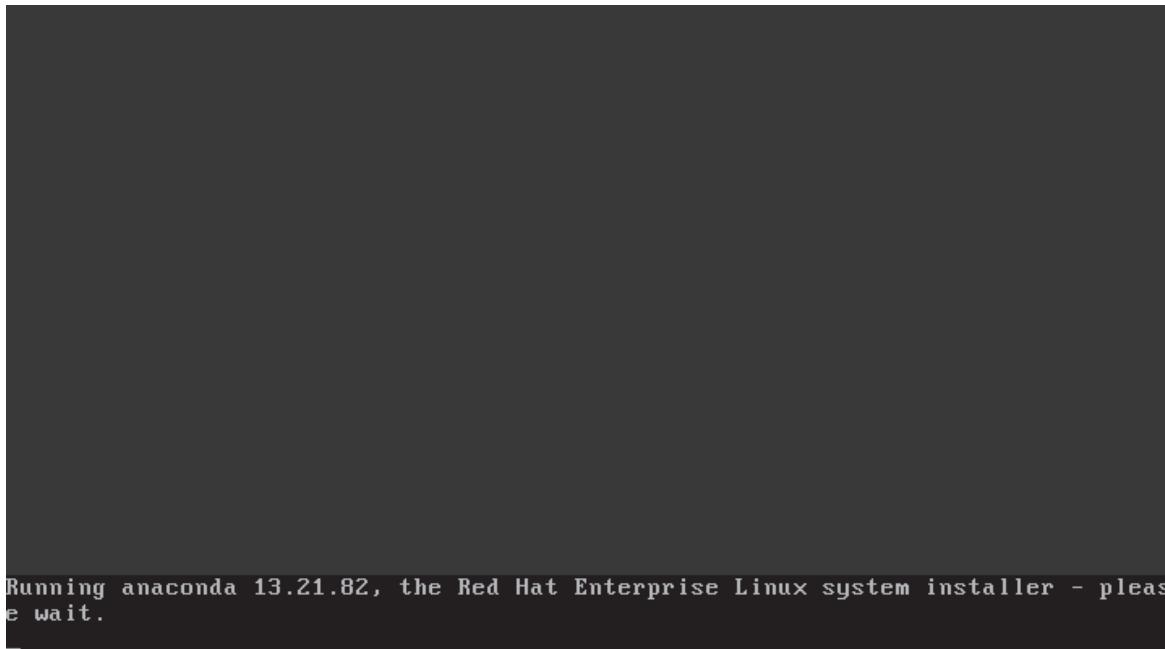


Figure 3.4: Running Anaconda Screen

Step 4 - A blank screen appears for a few seconds and then the graphical user interface of the installation appears as shown in Figure 3.5. Click Next.

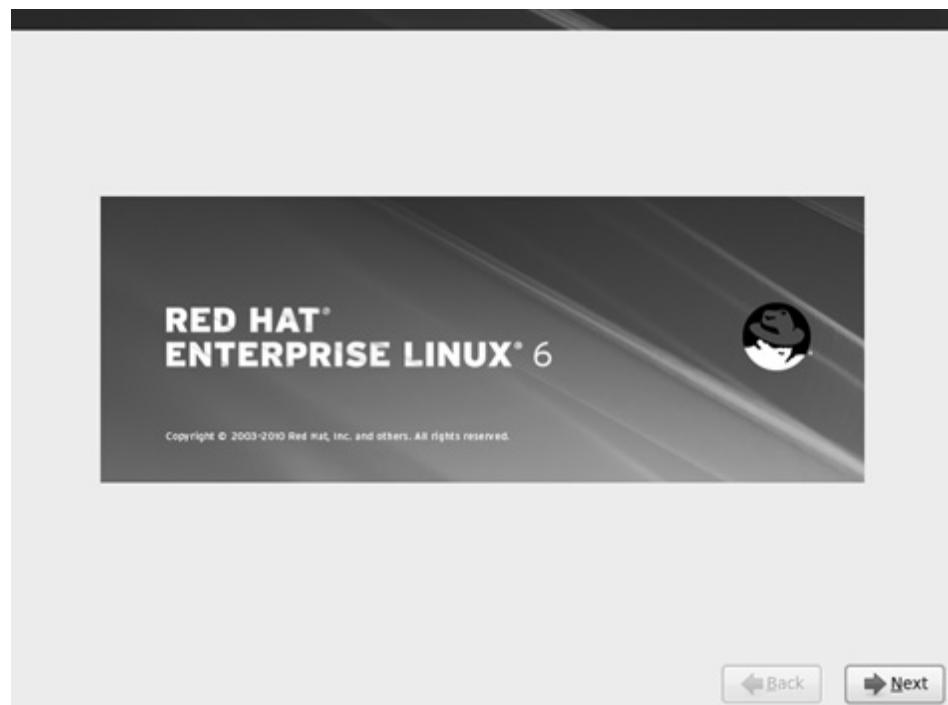


Figure 3.5: Graphical User Interface Installation Screen

Step 5 - Select the default language as English and click Next (Refer to Figure 3.6).

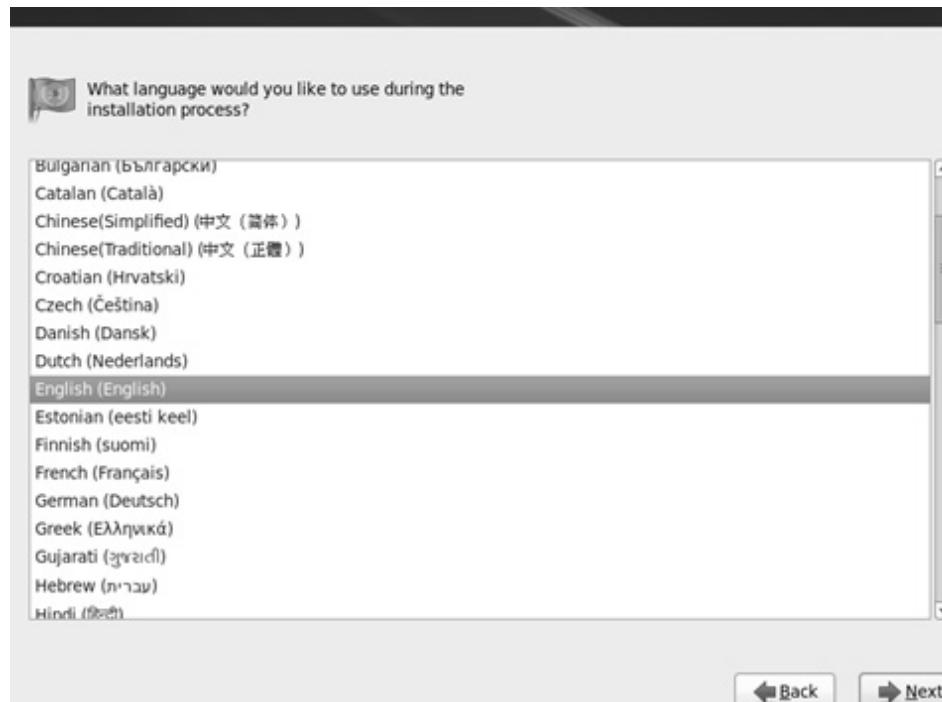


Figure 3.6: Selecting the Default Language

Step 6 - Select the appropriate keyboard as U.S. English and click Next (Refer to Figure 3.7).



Figure 3.7: Selecting the Keyboard

Step 7 - Click Basic Storage Devices and click Next (Refer to Figure 3.8).



Figure 3.8: Selecting Type of Device

Step 8 - On the Warning dialog box (Refer to Figure 3.9), click Re-initialize.



Figure 3.9: Warning Dialog Box

Step 9 - On the next screen, click Configure Network (Refer to Figure 3.10) to verify the network settings.



Figure 3.10: Configure Network Option

Step 10 - The Network Connections dialog box is displayed (Refer to Figure 3.11). Click Close and then click Next.



Figure 3.11: Network Connections Dialog Box

Step 11 - Select the city and click Next (Refer to Figure 3.12).



Figure 3.12: Selecting the City

Note - Users can configure the time zone configuration even after the installation is completed. Use the Time and Date Properties tool to configure the time zone after installation is complete.

Step 12 - On the next screen, provide the root password, confirm the root password, and then click Next (Refer to Figure 3.13).



Figure 3.13: Providing the Root Password

Note - Users are prompted with an error if they leave the Root Password and Confirm fields blank. If users have provided a simple password, they are prompted with a warning for weak password.

Step 13 - For selecting the type of installation, keep the default selection and click Next (Refer to Figure 3.14). Users can also select from other types of installation.



Figure 3.14: Selecting the Type of Installation

Step 14 - In the Writing storage configuration to disk dialog box (Refer to Figure 3.15), click Write changes to disk.



Figure 3.15: Writing storage configuration to disk Dialog Box

The creating of the file system starts as shown in Figure 3.16.



Figure 3.16: Creating File System

Step 15 - On the next screen, click Desktop and click Next (Refer to Figure 3.17).

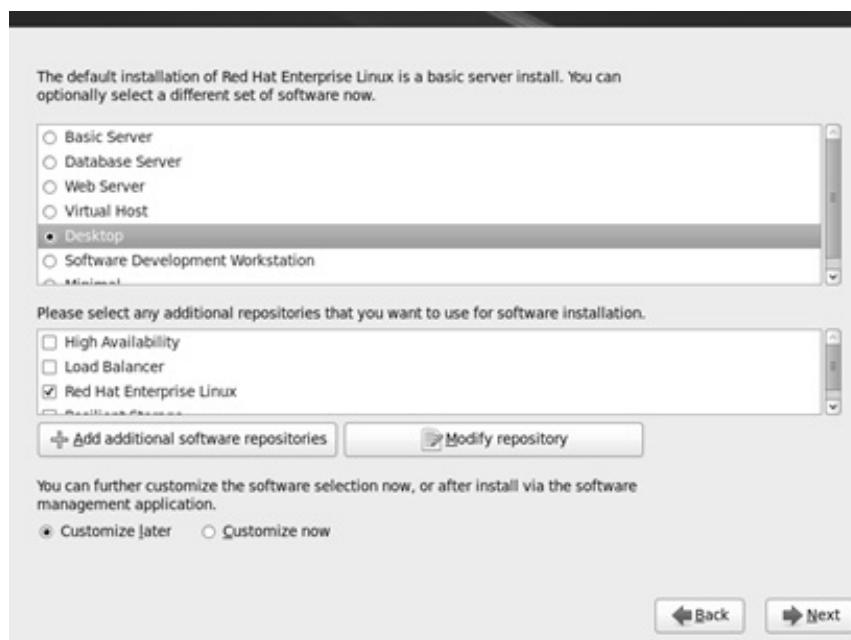


Figure 3.17: Selecting the Component for Installation

The installation process checks for the dependencies for the packages that have been selected for installation (Refer to Figure 3.18).

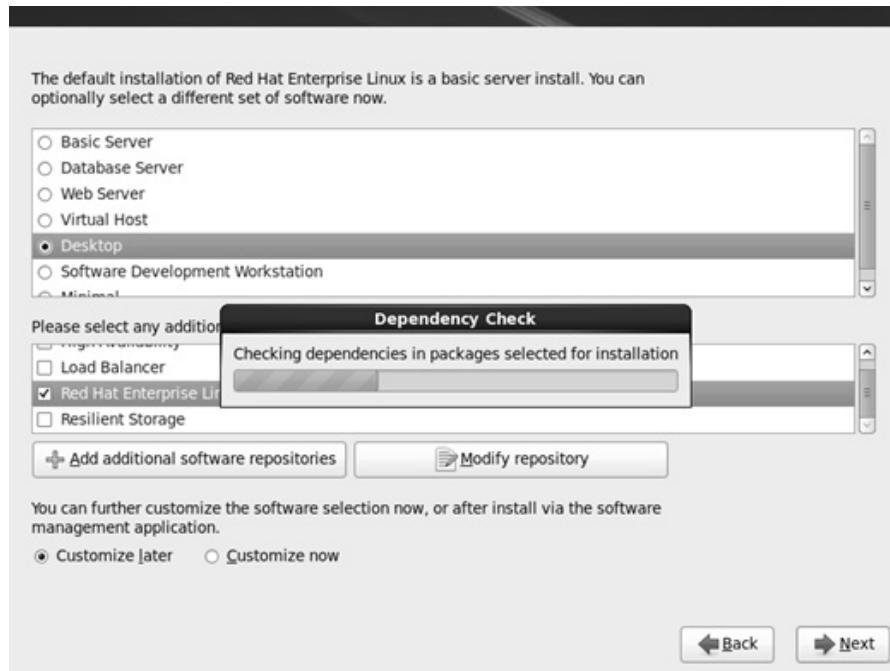


Figure 3.18: Checking for Dependencies

Note - Users can customise the installation of packages even after the installation is complete. The installation moves to the next stage of transferring the image to the hard disk and then initiating the installation process.

Formatting of file system is initiated before the operating system files can be copied on to the file system (Refer to Figure 3.19).



Figure 3.19: Copying File

After formatting is completed, the installation process starts. (Refer to Figure 3.20)



Figure 3.20: Installation Process

The installation process (Refer to Figure 3.21) starts to install the components. It takes approximately 60 minutes to complete the installation. However, the time can vary due to the machine configuration and packages selected for installation.



Figure 3.21: Component Installation Process

Step 16 - Click Reboot (Refer to Figure 3.22), to complete the installation and restart the system.



Figure 3.22: Reboot Button

Step 17 - After the system restarts, the users are prompted with the Welcome screen (Refer to Figure 3.23). Click Forward.

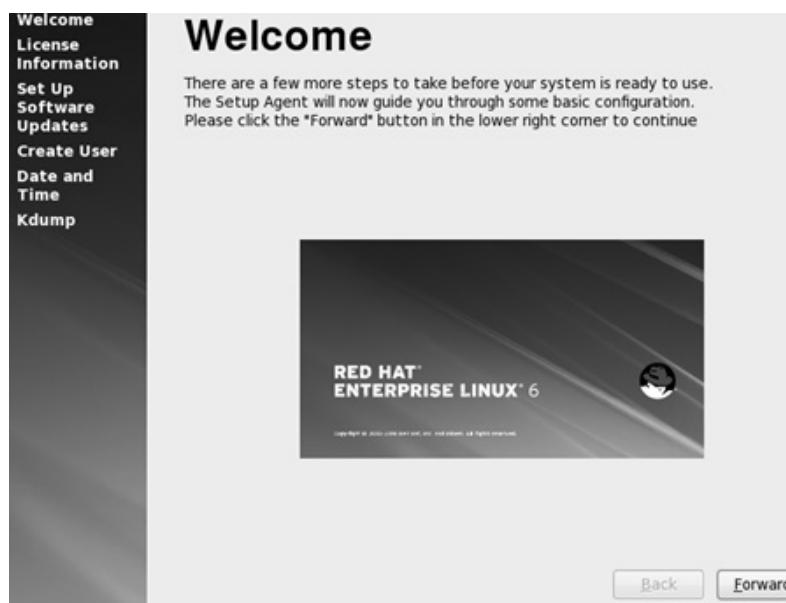


Figure 3.23: Welcome Screen

Step 18 - On the License Information screen (Refer to Figure 3.24), click Yes, I agree to the License Agreement, and then click Forward.



Figure 3.24: License Information Screen

Step 19 - On the Set Up Software Updates screen (Refer to Figure 3.25), click Forward.



Figure 3.25: Set Up Software Updates Screen

Step 20 - On the Finish Updates Setup screen, click Forward (Refer to Figure 3.26).



Figure 3.26: Finish Updates Setup Screen

Step 21 - On the Create User screen (Refer to Figure 3.27), provide the details and click Forward.



Figure 3.27: Create User Screen

Step 22 - On the Date and Time screen (Refer to Figure 3.28), select Synchronize the date and time over the network or manually set the time, and then click Forward.



Figure 3.28: Date and Time Screen

Step 23 - On the Kdump screen, click Finish.

Step 24 - The post installation configuration is completed. After the system reboots, users are prompted with the user authentication to log on to the system. Provide the user credentials and click Log In (Refer to Figure 3.29)

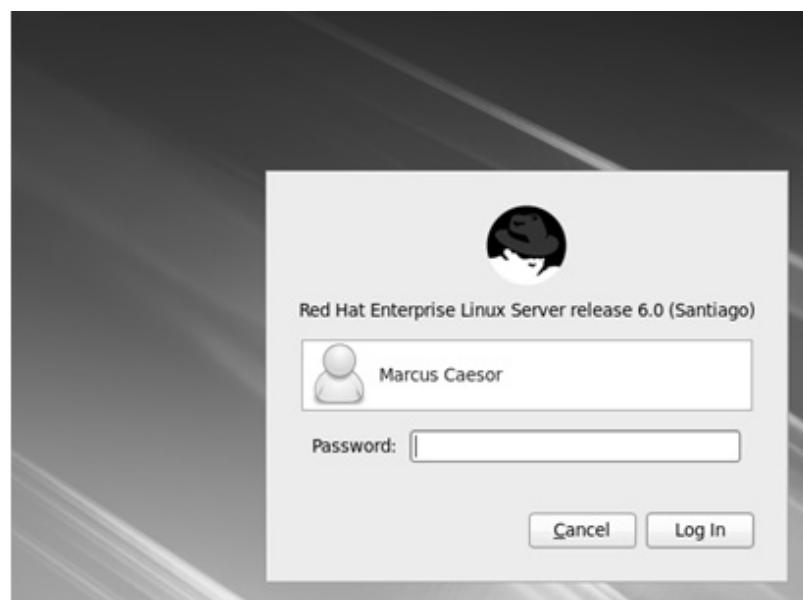


Figure 3.29: Providing User Credentials

After successful log in, the RHEL operating system desktop appears. (Refer to Figure 3.30)

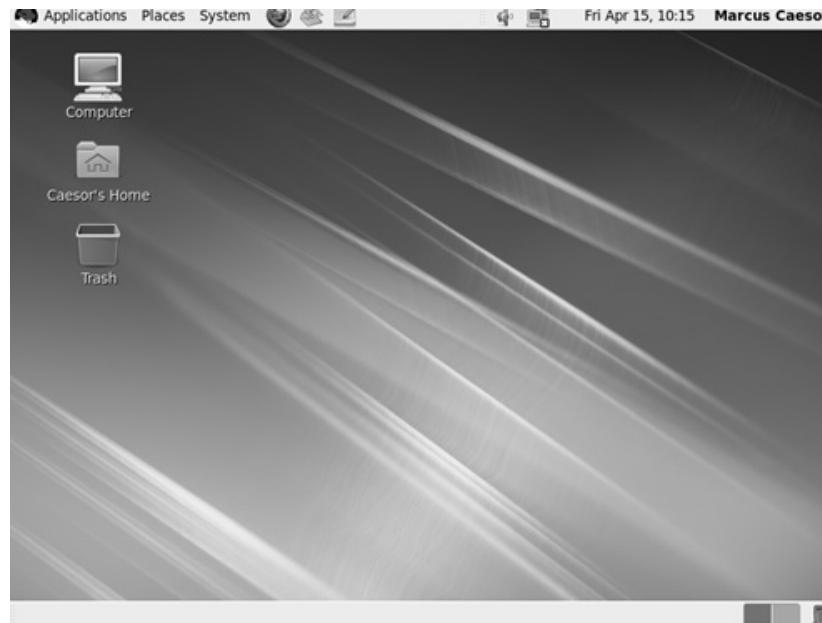


Figure 3.30: RHEL Desktop

Exercise - 2

Accessing the command line from Menu and Desktop

Step 1 - Log on to the RHEL workstation using the username and password.

Step 2 - Click Applications→System→Tools Terminal to open the command-line terminal, as shown in Figure 3.31.

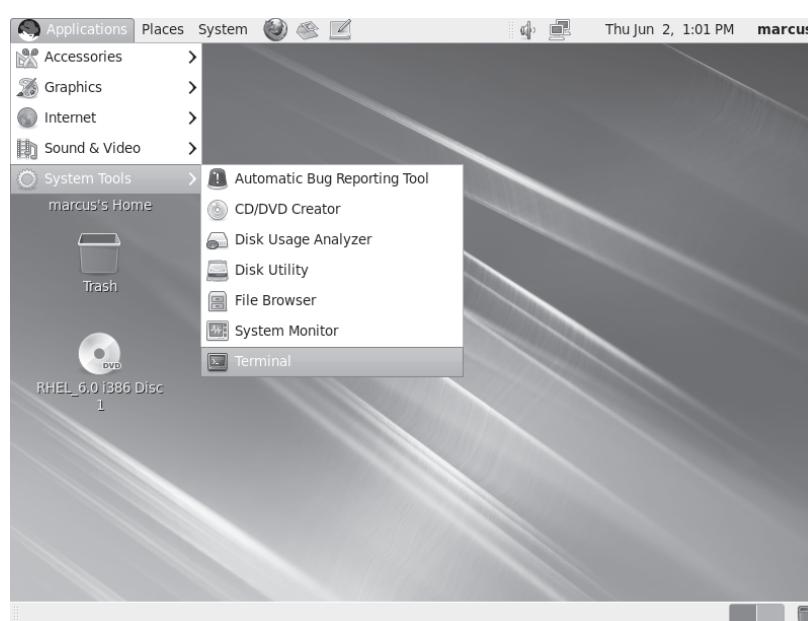


Figure 3.31: Navigating to Command-Line Terminal

The command-line terminal appears as shown in Figure 3.32.

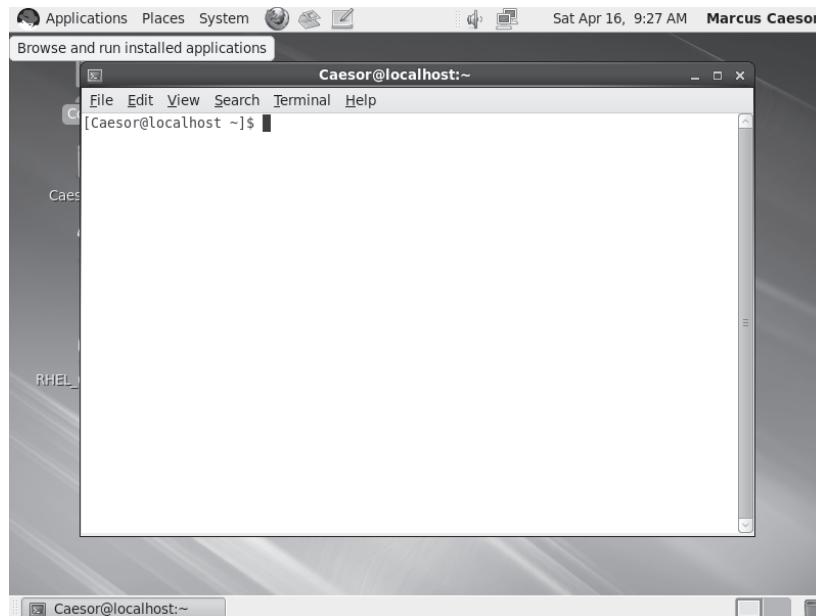


Figure 3.32: Command-Line Terminal

Step 3 - Type exit to close the terminal session and go back to the desktop.

Step 4 - Right-click the desktop and click the Open in Terminal option. This also opens the terminal session as shown in Figure 3.33.

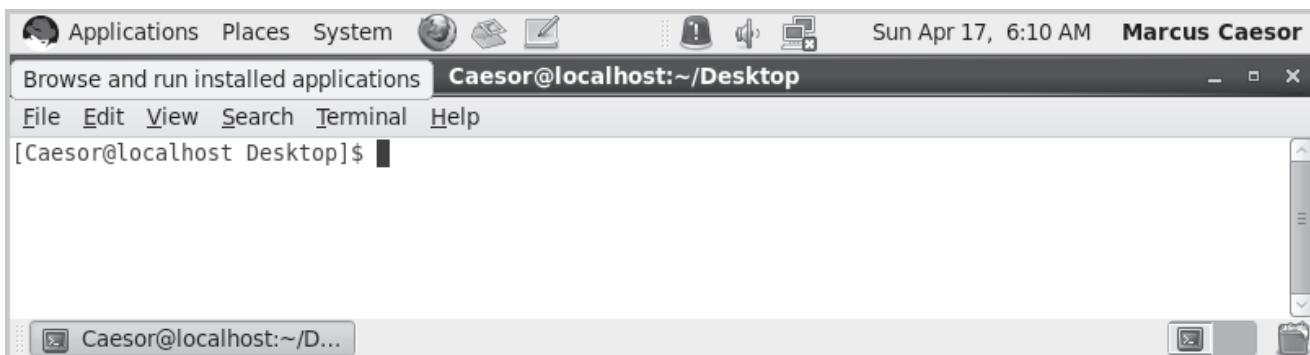
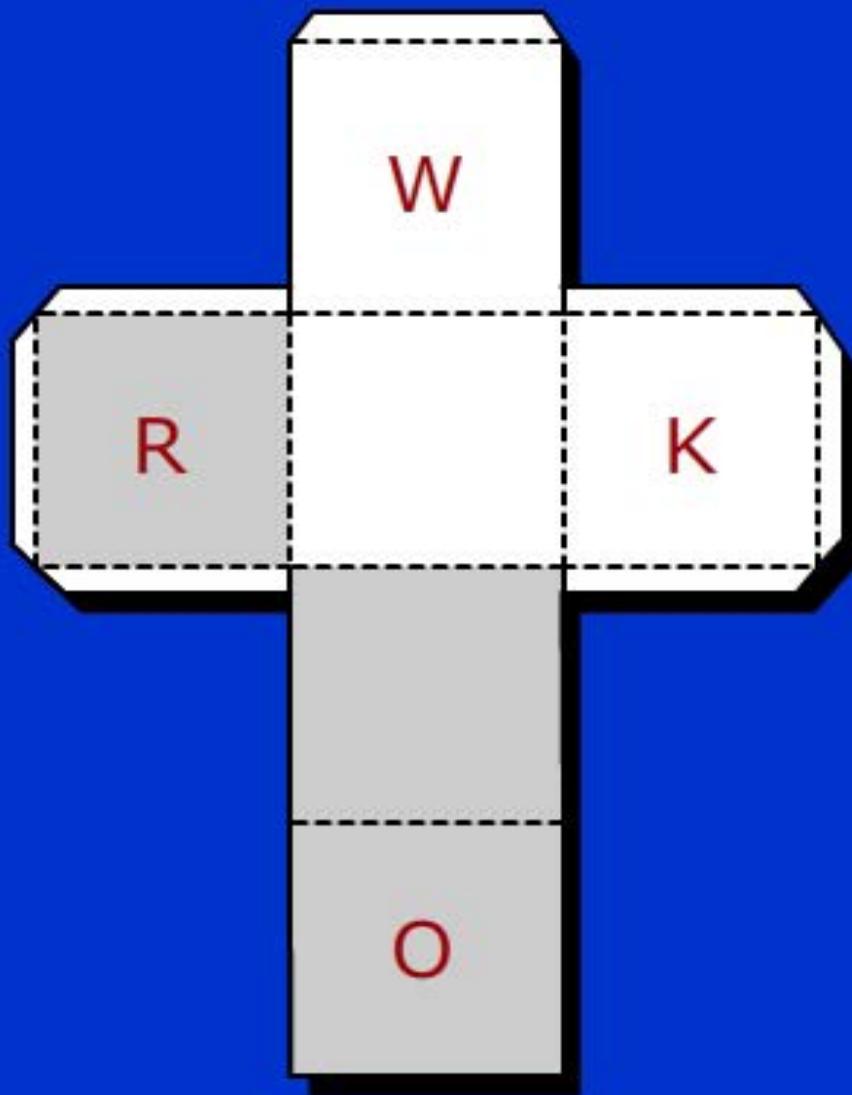


Figure 3.33: Command-Line Terminal from Desktop

Note - The only difference in invoking from the menu or desktop is the directory that is accessed from the command-line. When the user opens the terminal from the desktop, the localhost Desktop directory is opened. However, when the user opens the terminal from the menu, the localhost directory is opened, which contains many directories including Desktop.

WORK



ASSIGNMENT:

Form the cube to read '**WORK**'.

"Practice does not make perfect. Only perfect practice makes perfect."

- Vince Lombardi

Practice the Practicals for Perfection @

www.onlinevarsity.com

4.1 Understanding Command Line from X

The X Window System is based on the client-server architecture that includes two primary components, X server and X client. The X server, also called the XFree86 server, is a program that enables various applications, called the X clients, which run on a Linux system to display their GUI components.

The X server and the X client can run on same or different systems. The X clients send the display request to the X server, which in turn, interacts with the various hardware devices of the system to respond to the request of the X client. In addition, the other important functions performed by the X server are as follows:

- Displays the errors that occur when the X client makes a request for display
- Controls the I/O devices, such as the keyboard, mouse and display device
- Directs the keyboard and mouse input to the desired X clients or X events on the network
- Manages the windows displayed on the screen by creating, mapping and closing the windows

The key functions performed by X client are as follows:

- Requests the server for various services
- Accepts events in response to the requests from server
- Accepts errors messages from the server

The following example gives an insight on the client-server architecture of X Window System. When a hardware event, such as a mouse action, occurs, the X server receives the information. The server passes this information to the client application. The client processes the received information and sends a request to the server. An example of a request can be to display a character or to draw some graphics on the screen. The server executes the request.

The usefulness of X Window System cannot be experienced if the client and the server are present on the same computer. However, the potential of the client-server architecture is realized when it is used across the network on multiple computers. The advantages of using the client-server architecture of X Window System are as follows:

- The user can install the client and server software on different computers.
- The user can port the X client applications to other platforms.
- There can be multiple X servers running on more than one computer at a time.

4.1.1 Accessing Command Line from X

A user can access command-line in the GUI mode by using a terminal. A terminal is an instance of a command-line shell accessed from the GUI mode.

A user can create multiple terminals as per the requirement to execute commands. To create a terminal from the GUI mode, select **Applications→Accessories→Terminal** from the GNOME desktop.

Users can also invoke a new instance of the GNOME terminal by executing the following command from a GNOME terminal.

```
# gnome-terminal
```

4.1.2 Working with Linux Shell

Shell is a term used for the interactive user interface of an operating system. It is the layer of programming that understands and executes the commands entered by users. Shell script is a file that contains the sequence of commands to perform a particular task or operation on the computer. Execution of such shell scripts saves effort and time. Shell allows the users to handle a system with flexibility and to automate tasks in the form of shell scripts.

In a multiuser environment, the shell has to isolate the user interface from the kernel. This is because the kernel has to distribute the processing time of the operating system efficiently among the different active applications. It is pertinent to note that the shell acts as an interface between the user and the kernel and it plays an important role in the command interpretation for the kernel.

The user does not directly interact with the hardware. Instead, the kernel is in charge of the user interaction with the hardware, scheduling tasks and allocating resources to the running processes. Each user has access to the shell to work with the kernel.

Figure 4.1 shows the complete representation of the functioning of a Linux system that explains the importance of shell and its role in the system.

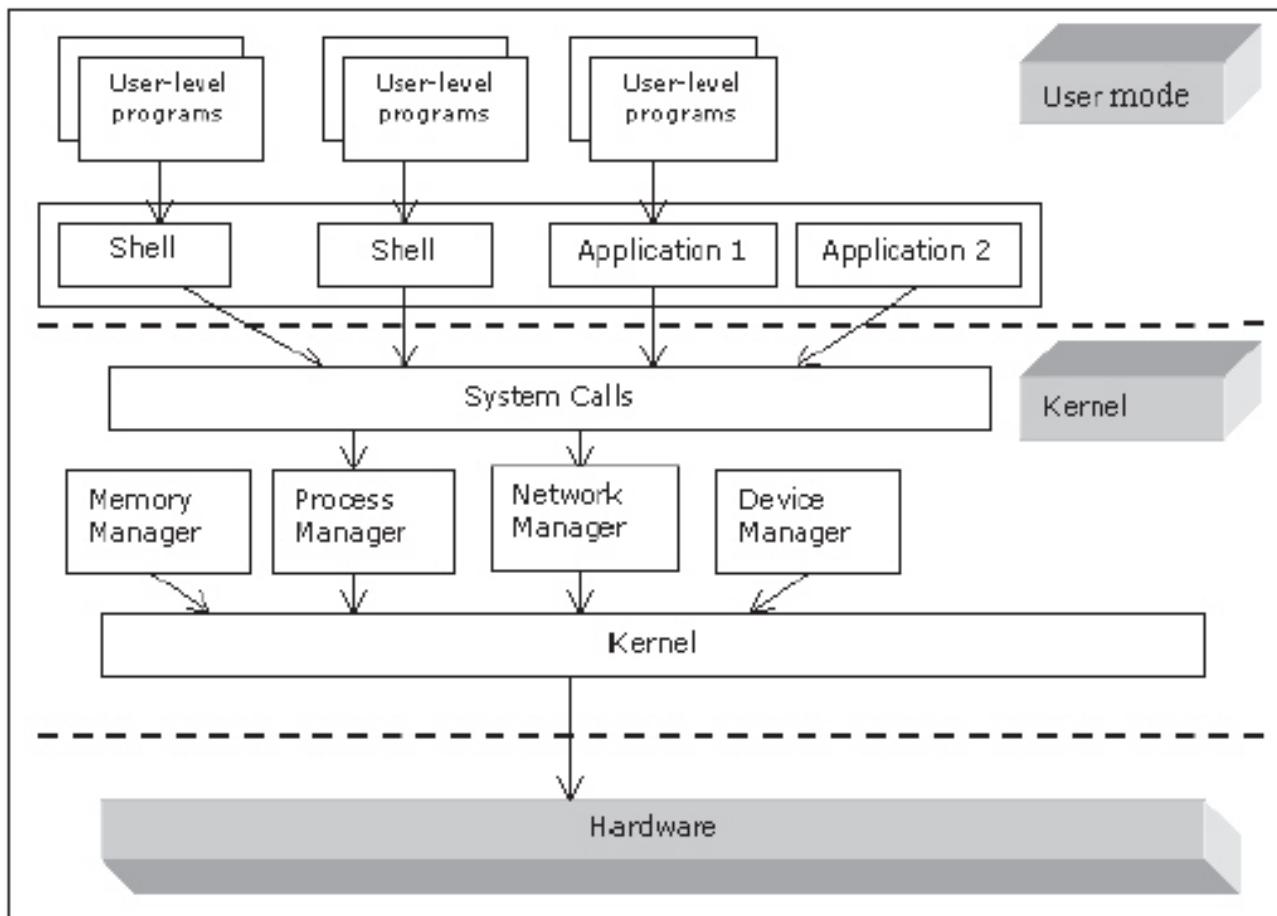


Figure 4.1: Functioning of the Linux System

Figure 4.1 shows that the shell interacts with the kernel through the system calls, which are a set of routines that allow an application to access kernel services. Different applications or daemons, manage each of the services, such as memory, processes, network, I/O devices and file and directory services. Daemons run at the kernel level and are always active. An example of a daemon is the cron daemon that is used to run commands, scripts or programs at scheduled times.

The shell on Linux can be compared to the command.com file in MS-DOS that provides similar functionality. The command.com file is one of the most important files of MS-DOS because it is the default operating-system shell and command-line interpreter. Shells are called command processors or command interpreters in MS-DOS. In a Microsoft Windows platform, such as Windows NT, cmd.exe is a shell.

Login and Non-Login Shells

Login shells are interactive shells that set up a certain pre-defined environment when a user logs on. On the other hand, non-login shells do not set any pre-defined environment and they inherit the environment of the parent shell under which they are created.

For example, the logon screen to specify the user id and password, when a shell is starting, is called a login shell. Login shells include the shells invoked using the su command. Login shells also set up a pre-defined user environment before being launched.

On the other hand, non-login shells are created when users execute a command or a shell script, invoke a graphical terminal or run the bash or sh command.

4.2 GRUB Boot Loader

Consider a scenario where a number of trainees have joined XYZ Inc. to gain experience on Linux administration. Steve, the system administrator at XYZ, provides a system installed with Linux to each trainee. One of the systems is unable to perform the boot operation. To repair the system, Steve has to boot the system using a bootable CD and then perform the diagnostic tests and fix the issues.

The booting process is the first process that is executed when a Linux system is started. This process of execution is as follows:

1. The processor searches for the Basic Input/Output System (BIOS) program in the memory and executes it. BIOS is a program that provides the lowest-level interface to the devices, such as hard disk, monitor and mouse.
2. BIOS runs the Power-On Self Test (POST) program that checks the system hardware by verifying the hardware configuration information. This is done to ensure that all the required hardware devices are present and are functioning.
3. BIOS checks the system memory for any hardware-related errors and searches for a bootable device. The bootable device can either be the hard disk or other bootable devices, such as a CD-ROM or a disk drive. The sequence in which BIOS searches for the various devices for the boot record is called a booting sequence.

Specify the booting sequence by modifying the booting sequence information in the BIOS setup program. To modify the BIOS setup program, press the <F1> or <Delete> key on the first startup screen. The boot sequence can be specified as follows:

CD-ROM, C, A

Where C represents the hard drive and A represents a network drive or Universal Serial Bus (USB) drive.

In the preceding boot sequence, BIOS first searches the CD-ROM for the boot record, then the hard drive and in the end the disk drive.

Note - The key used to access and customise the BIOS settings of a system depends on the vendor of the system. In addition, the boot sequence specification of the BIOS depends on the vendor of the system.

BIOS checks the bootable device for the presence of the Master Boot Record (MBR), which is the first sector of the bootable device. The MBR is 512 bytes in size and consists of a boot loader and a partition table. The boot loader contains instructions in the form of machine code, for booting the

system. The partition table contains information about the various partitions of the storage devices, such as the size of various partitions and names of the partitions.

4. BIOS locates and invokes the boot loader by passing the control of the system to the Initial Program Loader (IPL), also called the first stage of the boot loader, which is present in the MBR.
5. IPL loads the boot loader into the memory of the system. The boot loader takes the control of the system from the IPL when it is loaded into the memory of the system.

4.2.1 Understanding GRUB Boot Loader

A boot loader is a program that loads the operating system into the system memory while booting. The program is made up of two parts, primary boot loader and secondary boot loader. The part of the boot loader that is loaded into the memory of the system by BIOS at the start of the booting process is called the primary boot loader. The part of the boot loader that is loaded into the memory of the system by the primary boot loader is called the secondary boot loader.

The secondary boot loader is either directly loaded into the memory or an intermediary stage, called stage 1.5 boot loader. This is loaded into the memory before the secondary boot loader. The stage 1.5 boot loader can be present either on the MBR or on the boot partition. The boot partition is a partition on the hard disk that contains the files required for booting. The stage 1.5 boot loader is loaded into the memory before the secondary boot loader if the boot partition is present beyond the 1,024 cylinder heads of the hard drive. The stage 1.5 boot loader then loads the secondary boot loader, also called the stage 2 boot loader, into the memory of the system. However, if the boot loader is present within the 1,024 cylinders of the hard disk, the stage boot loader is loaded into the memory directly.

The secondary boot loader allows accessing the boot loader menu interface. This interface is used to customise the booting process by modifying the boot-loader commands and specifying the parameters to be passed to the kernel at the time of booting.

Using GRUB

GRUB is the default boot loader in RHEL. This boot loader enables to select an operating system from the various operating systems installed on the computer at the time of booting. GRUB displays a menu interface that lists the various operating systems installed on the system. To select an operating system from the list, use the UP and DOWN arrow keys and then press the ENTER key.

Features of GRUB

All the recent versions of the Linux operating system support the GRUB boot loader. This is because GRUB offers the following advantages:

- **Direct and Chain-Loading Booting** - Supports the direct as well as the chain-loading process of booting a system.

In the direct booting process, the boot loader of the operating system directly boots the system. However, in the chain-loading booting process, the boot loader of the operating system does not directly boot the system. Chain-loading booting takes place when the actual boot loader required

to boot the operating system is not present in the MBR. Instead, any other boot loader, which is present in the MBR, is invoked. This, in turn, invokes the actual boot loader of the operating system.

Linux follows the direct booting process, whereas, MS-DOS and Windows follow the chain-loading booting process. As GRUB supports both the booting processes, it can boot Linux as well as the Windows operating systems.

- **Logical Block Addressing (LBA) Mode** - Allows GRUB to boot the operating system from any of the memory addresses, which is either within the first 1,024 cylinders of the disk or beyond. In the LBA mode, the actual boot loader files can be present anywhere on the hard disk but their logical addresses are stored in the first 1,024 bytes of the memory. Therefore, GRUB does not have the 1,024 cylinders limitation.
- **Command-Based, Pre-Operating System Environment** - Allows the user to pass parameters to the kernel to configure the system according to the requirements.
- **ext2 Partition Support** - Eliminates the requirement of updating stage 1 of the boot loader on the MBR each time the configuration of the system changes. Therefore, reinstall GRUB on the MBR only if the physical location of the boot partition is changed manually.
- **Menu Interface** - Allows to install different operating systems and to customize the booting process by passing parameters to the kernel at the time of booting.
- **Flexible Command-Line Interface** - Allows execution of any GRUB command at the boot time to customize the booting process.
- **Automatic File Decompression** - Decompresses the zipped file by automatically using the gzip utility. It reduces the overall loading time of the file.
- **Network Booting** - Allows the loading of the operating system image from the network by using the Trivial File Transfer Protocol (TFTP). An operating system image is an image file that contains the information about the operating system files.

GRUB Interfaces

The GRUB interfaces enable the customization of the booting process. The various interfaces provided by GRUB are as follows:

- **Menu Interface** - This interface appears by default, if the Linux installation program configures GRUB. The menu interface is also called the graphical splash screen. It displays a list of various operating systems that can be installed using GRUB in the alphabetical order. The user can select a particular operating system from the list. If the user does not select an operating system within a specified time that is the time-out period, GRUB loads the operating system set as default.

- **Menu-Entry-Editor Interface** - This interface is used to edit the GRUB commands that are required to be executed while booting. To switch to this interface, press the 'e' key on the menu interface. This interface also shows information, such as the name of the boot partition and the kernel number, which is required by the GRUB at the time of booting. The various commands to modify the GRUB configuration file are also displayed in the menu-entry-editor interface.
- **Command-Line Interface** - This interface is used to execute the GRUB commands by providing a command-line. To switch to the command-line interface, press the 'c' key on the menu-entry-editor interface or the menu interface. Type a GRUB command in this interface and press the ENTER key to execute the command.

GRUB provides various command-line interface commands. These commands are used to customize the booting process for a particular session. Conversely, the commands specified in the GRUB configuration file are executed each time the system boots. Table 4.1 describes the various GRUB commands.

Command	Description
boot	Boots the operating system or the chain loader that has been specified.
chainloader </path/to/file>	Loads the specified file as a chain loader. If the file is located on the first sector of the specified partition, use the block list notation instead of the filename.
displaymem	Displays the amount of RAM that a system has prior to booting.
initrd </path/to/initrd>	Specifies an initial RAM disk, initrd, to be used while booting. An initrd is necessary when the root partition is formatted using the ext3 file system.
install <stage-1><install-disk><stage-2><config-file>	Installs GRUB in the MBR. The parameter, <stage-1>, signifies a device, partition or file where the primary boot loader is stored. The parameter, <install-disk>, specifies the disk where the primary boot loader should be installed. The parameter, <stage-2>, specifies the location of the secondary boot loader to the primary boot loader. The parameter, <config-file>, specifies the path of the GRUB configuration file.

kernel </path/to/kernel><option-1> <option-N>	Specifies the kernel file to be loaded while booting the operating system. The parameter, </path/to/kernel>, specifies the path of the root command. The parameter, <option-1> <option-N>, specifies the options for the Linux kernel, such as root=/dev/ hda5, to indicate the device on which the root partition of the system is located.
root (<devicetype><devicenumber>, <partition>)	Configures and mounts the root partition, such as (hd0, 0), for GRUB.
rootnoverify (<devicetype><devicenumber>, <partition>)	Configures the root partition for GRUB but does not mount the partition.

Table 4.1: GRUB Commands and its Description

4.2.2 Identifying Different Runlevels

Runlevels represent the various modes in which a user can work on a Linux system. Running Linux in different modes helps utilize the computing resources efficiently for most of the important tasks at hand.

For example, Linux can be used in single-user mode where network services are not enabled so that computing resources are utilized for user-level computing tasks only. On the other hand, Linux can also be used in network mode where network services are available and personal user-level services are not available.

Runlevels also help in assigning privileges such as access to the network or GUI. For example, to restrict a user to access the network, configure Linux to run in single-user mode. Runlevels also helps to reduce the potential attack surface area of a Linux system, as they do not allow unnecessary services to run.

Each runlevel executes certain services, such as Sendmail and Apache, to enable the mailing and Web services, respectively. These services have symbolic links that are present in the rc directory that correspond to the runlevels. The symbolic links are the pointers to the actual scripts that are present on the hard disk. A user can create, modify and delete the symbolic links to specify the services that are to be started in a particular runlevel. Deletion of the symbolic links does not affect the scripts referenced by them.

Each symbolic link has a name that begins with the letters K or S. K specifies that the service is terminated and S specifies that the corresponding service is started in a runlevel. The symbolic links point to the actual scripts present in the /init.d directory. The scripts that correspond to a particular runlevel are executed when the user switches to that runlevel. Each script is associated with a service that must run in a particular runlevel. When a service starts, it generates various processes each of which is assigned a Process ID (PID) by the operating system.

There are System V Initialization (SysV init) runlevels that are used to control the services that are started or halted by the init program at the time of initializing a runlevel. The SysV init runlevel system provides a standard process for controlling the programs that are launched or halted by the init, while initialising

a runlevel. The configuration files for SysV init are located in the /etc/rc.d/ directory. Linux defines six basic runlevels, 0, 1, 2, 3, 5 and 6. The system boots in one of these six runlevels. The default runlevel is specified in the /etc/inittab directory of the system.

To boot the system in a particular runlevel other than the default runlevel, modify the GRUB configuration file and reset the value of the default runlevel when user boot the system. This action does not change the default runlevel settings permanently and allows to boot the system in the runlevel that has been specified for that particular session. The runlevel that is chosen to boot the system depends on the task to be performed.

Table 4.2 describes the various runlevels defined by Linux.

Runlevel	Description
0	Halts the Linux system.
1	Starts the Linux system in the single-user text mode.
2	Starts the Linux system in the full multiuser text mode without enabling a networking service. In this case, the multiple users can log on to the system at a time and work in the command-line interface mode.
3	Starts the Linux system in the full multiuser text mode with the networking service enabled.
4	Allows the user to define this runlevel.
5	Starts the Linux system in the full multiuser graphical mode.
6	Reboots the Linux system.

Table 4.2: Runlevels

Note - Set the default runlevel value to 0 or 6. If the default runlevel value is set to 0, the system halts each time the booting process starts. If the default runlevel value is set to 6, the system always restarts whenever the user tries to boot the system.

Working with Runlevels

Users require different services while working on different runlevels. These levels can be configured by enabling and disabling these services. Linux provides certain utilities to configure the different runlevels.

The utilities offered by Linux are as follows:

- **Chkconfig Utility** - Adds, removes and manipulates the various services that should be started in a particular runlevel. It is a command-line utility for maintaining the different runlevels in the /etc/rc.d directory.

In addition, the user can check the status of a particular service whether it is enabled or disabled.

To check the status of service use following command syntax:

```
# /sbin/chkconfig --list <service_name>
```

where,

<service_name> parameter is the name of the particular service that the user has to check the status

Example:

To check the status of sendmail service that is used to send and receive mails in different runlevels use following command:

```
# /sbin/chkconfig --list sendmail
```

The output of the preceding command is as follows:

```
sendmail 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

To turn off a service for a particular runlevel, use the following command syntax:

```
/sbin/chkconfig --level <level_number> <service_name> <on|off|reset>
```

where,

<level_number> parameter is used to specify the runlevel for which the user requires to turn off the service specified by the parameter <service_name>. The other parameters, on and off are used appropriately, when the service must be turned on or off respectively. The **reset** parameter is used to reset the runlevels for the service.

Example:

To turn off the Sendmail service for runlevel 4, use the following command:

```
# /sbin/chkconfig --level 4 sendmail off
```

The output of the preceding command is as follows:

```
sendmail 0:off 1:off 2:on 3:on 4:off 5:on 6:off
```

To turn on the service again, replace the keyword ‘off’ at the end of the command with the keyword ‘on’ and execute the command.

- **The ntsysv Utility** - Configures the current runlevel by default and provides an interactive text-based interface. However, the user can use this utility to configure any runlevel by specifying the runlevel, using the – level flag. To start the **ntsysv** utility, use the command, **ntsysv**. The command displays the Services pane that lists the various services that can be configured in the runlevel in which the users are currently working.

Figure 4.2 shows the output of `ntsysv` utility.



Figure 4.2: Output of `ntsysv` Utility

- **Services Configuration Utility** - Configures various services in different runlevels by using an interactive graphical interface. Linux provides two types of GUIs such as KDE and GNOME. These GUIs are available only in runlevel 5.
- **Service Command** - Starts or stops stand-alone services that run independent of other services. The service command can be used with various arguments to customise its action.

The syntax of the **service** command is as follows:

```
service <Service_name> <arguments>
```

Example:

```
service sendmail start
```

This command starts the sendmail service.

The various arguments and their functions are listed in Table 4.3.

Argument	Function
start	Starts a particular service.
stop	Stops a particular service.
reload	Updates the configuration file of the service and implements the changes to the service without stopping the service.
restart	Restarts a particular service that is already running.
condrestart	Restarts the service when a particular condition is met.

Argument	Function
status	Shows the status of a particular service at a particular time.

Table 4.3: Arguments

4.3 Basic Commands

RHEL provides some basic commands and command line expansions that help in configuring the system. For example, users can view the current system date and time and modify the system screen using various commands, such as, **date**, **clear**, **tput cup** and **tput reset**. Some of the command-line expansions include

~, {} and \$.

The **date** Command

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

The syntax of the **date** command is as follows:

```
# date
Thu Aug 12 15:56:21 IST 2011
```

The display of the **date** and time can be formatted using the various options of the **date** command. These options must be enclosed in quotation marks and must begin with a '+' symbol.

Table 4.4 lists the options available with the **date** command.

Option	Description
%m	Displays the month (in digits)
%d	Displays the day (in digits)
%y	Displays the year (last two digits)
%D	Displays the date as mm/dd/yy
%H	Displays the hour (00 to 23)
%M	Displays the minutes (00 to 59)
%S	Displays the seconds (00 to 59)
%T	Displays the time as HH:MM:SS
%a	Displays the abbreviated weekday (Sun to Sat)
%h	Displays the abbreviated month (Jan to Dec)
%r	Displays the time in the A.M./P.M. notation

Table 4.4: Options of the **date** Command

Example:

```
# date "+%T"  
22:47:45  
# date "+%y"  
11
```

Messages can also be included within the **date** command.

Example:

```
# date "+DATE : %D"  
DATE : 03/22/11
```

Note - The **date** command can also be used to change the system date and time. Only the system administrator has the rights to change the system date and time.

The clear Command

The **clear** command is used to clear the terminal screen. RHEL also allow users some measure of screen manipulation with the help of the following commands:

- ➔ **tput clear** - This command clears the standard output device and positions the cursor at the top left corner of the screen.
- ➔ **tput cup** - This command, followed by the screen coordinates, positions the cursor at the specified row and column.

Example:

```
tput cup 15 20
```

This command positions the cursor at row 15, column 20, as shown in Figure 4.3.

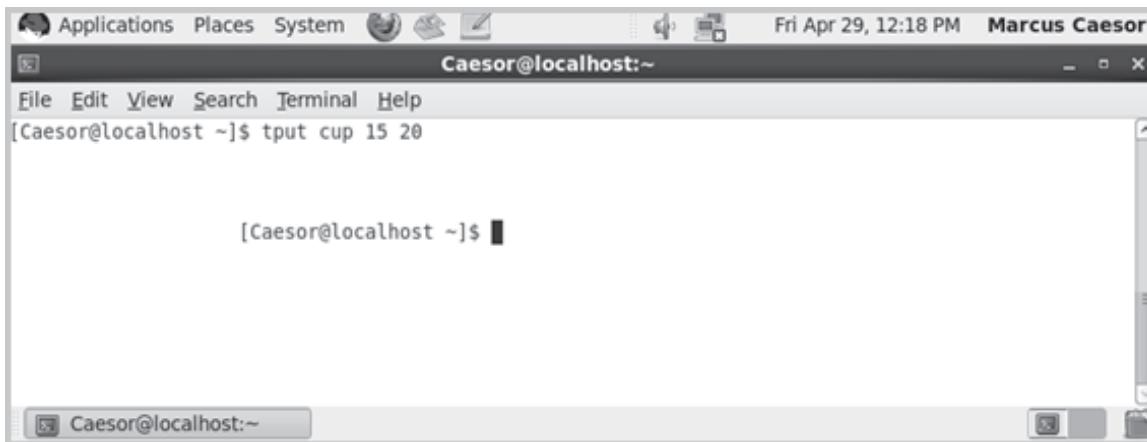


Figure 4.3: Positioning the Cursor on the Terminal Screen

- **tput smso** - This command sets the screen to reverse video.
- **tput rmso** - This command sets the screen back to normal.
- **tput blink** - This command displays a blinking output. Note that this option cannot work on a Telnet session.
- **tput reset** - This command resets the screen back to the default settings.

Some of the different command-line expansions are as follows:

- **~ (Tilde)** - The working directory of the user after logging on to the Linux system is the user's home directory. While working on Linux, a user can often require to access the home directory in a convenient way. In Linux, the path for the user home directory can be specified with the tilde (~) sign. For example, if Daniel has to access the September2011 file, he can specify the file name with the path as ~/baseball/ September2011 from any location in the file system.
- **\$** - The \$ character is used to substitute the value of a variable in a command line. For example, if users have to display value assigned to a variable, they are required to execute the following commands:

```
# var1=2
# echo "value of var1 is" $var1
```

The output of the command is 'value of var1 is 2'. Here, the \$ character is used to substitute the value assigned to the variable in the command line.

- **{}** - The {} are used to create string pattern of strings. Any command, where the {} are used, runs

once for each pattern mentioned inside the {}.

For example, the following command runs on two files file.c and file.java regardless of the existence of any such files:

```
# ls -al file.{c,java}
```

The ls -al command first runs and takes file.c as parameter and then it runs second time and takes file.java as parameter.

4.4 Identifying the Current Users Working on the System

The who command displays the names of users that are presently logged on to the system.

Example:

```
# who
root tty1 Sep 21 12:29
root tty2 Sep 21 14:54
sam tty3 Sep 21 14:57
aronpts/0 Sep 21 11:36 (172.17.55.178)
tom pts/1 Sep 21 11:15 (172.17.55.133)
sampts/2 Sep 21 22:14 (172.17.55.167)
marypts/3 Sep 21 22:16 (172.17.55.169)
```

Table 4.5 describes the various columns shown in the given example.

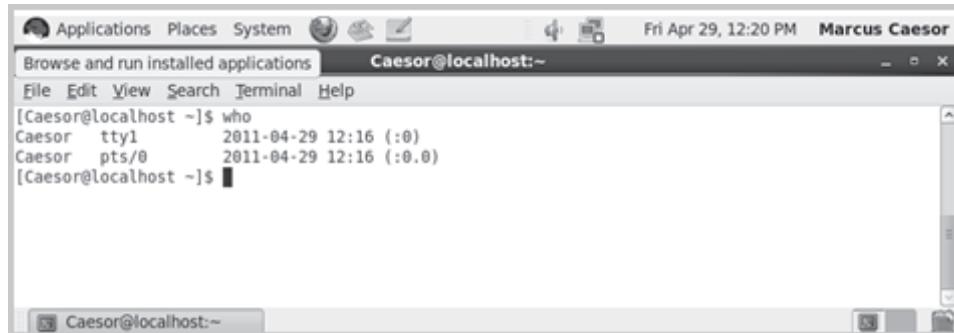
Column #	Description
1	Login name.
2	Terminal type and number.
3	Date and time of log on.
4	The remote host name of the terminal from where the user has logged on. Note that only the names of those users who have logged in from terminals other than the server are shown.

Table 4.5: Column Description of the who Command Output

The output of the who command also contains the terminal file name and the date and time the user logged on. In the preceding example, for the first three users, the terminal type is ttyN, where N is a number from 1 to 12. The pts terminal type is given only for users who have logged on from the server. It is possible for multiple users to simultaneously log on from the server using multiple terminals called virtual consoles.

Note - RHEL allows up to six users to log on to the operating system from the server. Virtual consoles can be invoked by pressing the <Ctrl><Alt><Fx> keys, where x is a number from 1 to 6 for the respective function keys. The <Alt> key and any of the function keys, when pressed can help to switch between the consoles.

Figure 4.4 shows the users that are logged on using the **who** command.



The screenshot shows a terminal window titled 'Caesor@localhost:~'. The window contains the following text:

```
[Caesor@localhost ~]$ who
Caesor  ttys1          2011-04-29 12:16 (:0)
Caesor  pts/0           2011-04-29 12:16 (:0.0)
[Caesor@localhost ~]$
```

Figure 4.4: Output of the **who** Command

The **who am i** command displays list of users that are currently logged on.

4.5 Using Filter

Filters are used to restructure the output based on a certain condition. In contrast to pipes, which help users to pass an output of one command as the input to the next command, filters help in taking the input, modifying it as per requirement, and then pass it as output.

The functions of filters are as follows:

- Takes its input from the standard input file
- Filters the input
- Sends the output to the standard output file

RHEL has filters that are used to work on data in an effective way. Some examples of filters are as follows:

- grep
- wc
- cut
- tr
- sort

The grep (Global Regular Expression Print) Filter

The **grep** filter is used to search and display the lines in a file with particular pattern of characters. The pattern that is searched in a file is referred to as the regular expression. The **grep** filter can be used only by specifying a regular expression.

The syntax for the **grep** filter is as follows:

```
# grep [options] pattern [filename]
```

File name is optional with the **grep** command. Without a file name, **grep** expects input from the standard input. After the user types a line, **grep** searches for the regular expression in the line and displays the line if it contains that particular expression. Execution stops when the user indicates end of input (by pressing the Ctrl+D keys).

Specifying Regular Expression

Regular expressions can be used to specify simple patterns of characters to highly complex ones. These characters are interpreted by the shell in a specific way. Symbols can also be used as a regular expression.

Table 4.6 lists the simple regular expressions.

Regular	Expression Pattern
'A'	The character A
'F'	The character F
'New'	The word New

Table 4.6: Simple Regular Expressions

Note - Regular expressions are always enclosed in single quotation marks.

Characters can also be used to specify complex regular expressions. Table 4.7 lists the complex regular expressions.

Character	Use	Example	Description
[]	Matches any one of a set of characters	grep "New[abc]"	It specifies the search patterns as 'Newa', 'Newb' or 'Newc'.
[] With Hyphen	Matches any one of a range of characters	grep "New[a-c]"	It specifies the search patterns as 'Newa', 'Newb' or 'Newc'.

Character	Use	Example	Description
^	The pattern following the ^ (caret) must occur at the beginning of each line	grep "^New[abc]"	It specifies the search patterns as 'Newa', 'Newb' or 'Newc', which must occur at the beginning of the line.
^ within []	The pattern that contains the word followed by any character other than the ones specified within brackets	grep "New[^a-c]"	It specifies search patterns that contain the word 'New', followed by any character other than 'a', 'b' or 'c'.
\$	The pattern preceding the dollar sign must occur at the end of each line	grep "New[abc]\$"	It specifies the search patterns as 'Newa', 'Newb' or 'Newc', which must occur at the end of the line.
. (dot)	Matches any one character	grep "New. [abc]"	It specifies patterns that contain the word 'New', followed by either 'a', 'b' or 'c'.
\ (backslash)	Ignores the special meaning of the character that follows the backslash	grep "New\.\\ [abc\\]"	It specifies a search pattern, New.[abc], where the dot signifies the dot character itself and [abc] signifies itself.

Table 4.7: Complex Regular Expressions

Options of the **grep** Filter

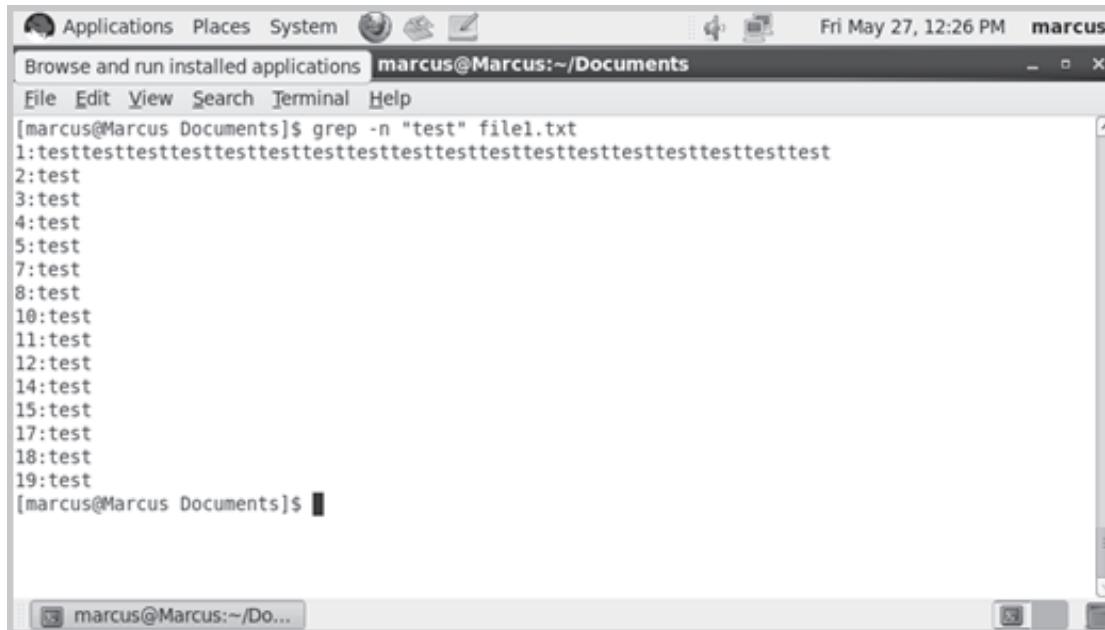
The **grep** filter also has options that alter the output of the command. The options are as follows:

- **-n** - Prints each line that matches the pattern, along with its line number. The number is printed at the beginning of the line.
- **-c** - Prints a count of the lines that match a pattern.
- **-v** - Prints all the lines that do not match the pattern specified by the regular expression.

Options must be specified before the regular expression. Options can also be combined.

For example, **-n** and **-v** options can be used together as **-nv**.

Figure 4.5 displays the result using the `grep` filter.



The screenshot shows a terminal window titled "marcus@Marcus:~/Documents". The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The status bar at the bottom shows "marcus@Marcus:~/Do...". The terminal content is as follows:

```
[marcus@Marcus Documents]$ grep -n "test" file1.txt
1:testtesttesttesttesttesttesttesttesttesttesttesttesttest
2:test
3:test
4:test
5:test
7:test
8:test
10:test
11:test
12:test
14:test
15:test
17:test
18:test
19:test
[marcus@Marcus Documents]$
```

Figure 4.5: grep Filter

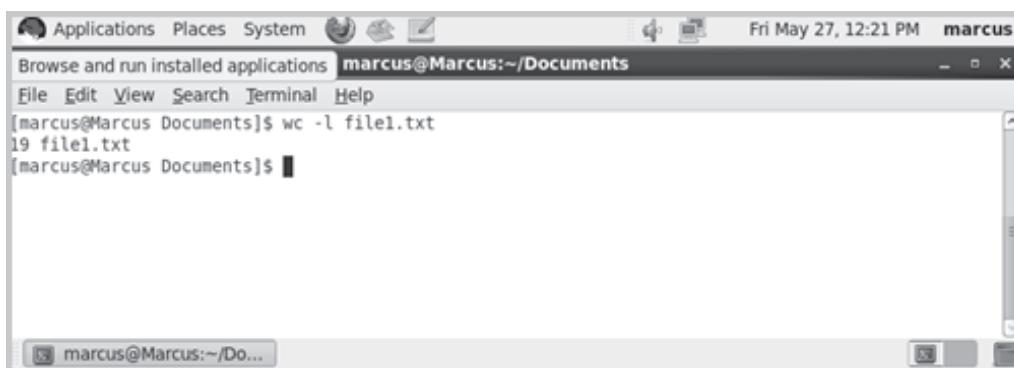
The `wc` Filter

The `wc` filter counts the number of lines, words and characters in a disk file or in the standard input file.

The syntax for the `wc` filter is as follows:

```
# wc [option] [filename]
```

The result of using the `wc` filter is displayed in Figure 4.6.



The screenshot shows a terminal window titled "marcus@Marcus:~/Documents". The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The status bar at the bottom shows "marcus@Marcus:~/Do...". The terminal content is as follows:

```
[marcus@Marcus Documents]$ wc -l file1.txt
19 file1.txt
[marcus@Marcus Documents]$
```

Figure 4.6: wc Filter

Table 4.8 lists the options of the **wc** filter.

Option	Description
-l	Used to display the number of lines
-w	Used to display the number of words
-c	Used to display the number of characters

Table 4.8: Options of the **wc** Filter

The **cut** Filter

The **cut** filter is used when specific columns from the output of certain commands such as **ls** and who have to be extracted. The syntax for the cut filter is as follows:

```
# cut [options] [filename]
```

Table 4.9 lists the options of the cut filter.

Option	Description
-f<column_number(s)>	Displays the specified columns
-c<character_number(s)>	Displays the specified character
-d<column_delimiter>	Specifies the column delimiter

Table 4.9: Options of the **cut** Filter

The **tr** Filter

The **tr** filter translates one set of characters to another. This filter can also be used to compress repeated occurrences of a character into one. Several commands have multi-column output and the gap between columns is more than one space. In such cases, if users have to extract a particular column, the cut filter cannot be used because the column separator has to be a single character. Therefore, users must replace the space between columns with a single space and then use the cut filter to extract the desired columns.

The syntax of the **tr** filter is as follows:

```
tr[Set1] [Set2]
```

Example:

```
tr [A-E][#]
```

The **sort** Filter

The **sort** filter is used to arrange each line of the standard input in ascending order. To use the command, users must type the **sort** command, and then enter the data that has to be sorted. Thereafter, users must press the **Ctrl+D** keys to sort the data.

The following example demonstrates the usage of the sort filter:

```
# sort
```

```
Sam
```

```
Brian
```

```
Daniel
```

```
Mike
```

```
Rodney
```

```
Andy
```

```
Sandy
```

```
Ctrl+D
```

```
Output:
```

```
Andy
```

```
Brian
```

```
Daniel
```

```
Mike
```

```
Rodney
```

```
Sandy
```

```
Sam
```

As shown in the example, the user types the names. After specifying the data, the user must press the **Ctrl + D** keys. The **sort** filter compares the characters on each line and displays the lines in the ascending order.

4.6 Using Pipes

Sometimes, writing multiple commands to perform a task is very tedious. For this, RHEL provides a feature called pipe that allows combining multiple commands to perform a task. Using pipes, users can direct the output of one command or filter to another command or filter as an input. This way users can combine two commands, which they have to run separately otherwise.

Following commands can be used to display the contents of the current directory, a screen-full at a time.

```
# ls > tempfile
```

```
# more tempfile
```

Here, the listing of the directory is stored in the file, **tempfile**, by the first command. This file is then used as input by the **more** command.

Note - The more filter is used to display text, one screen-full at a time.

Through the pipe feature, these two steps can be combined and executed as a single command, without creating a temporary file.

Example:

```
# ls | more
```

The vertical bar (|) is the pipe character, which indicates to the shell that the output of the command before the ‘|’ is to be sent as input to the command after the ‘|’.

Note - On the keyboard, the pipe character appears as a broken vertical line.

Another advantage of the pipe feature is that utilities are not rewritten to perform complex tasks. Instead, the RHEL tools (commands) can be combined. There is no limit to the number of filters or commands in a pipe.

Figure 4.7 shows the working of a pipe.

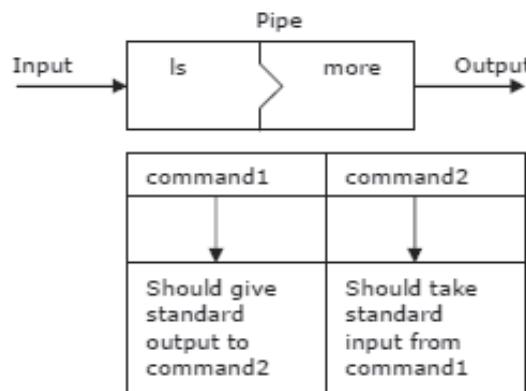


Figure 4.7: Working of a Pipe

The input is processed by the `ls` command, which is the first command for the pipe. After the input is processed, the output from the `ls` command is used as an input for the `more` command, which is the second command for the pipe. The pipe directs the output of one command to another command as input.

Some uses of the pipe feature are as follows:

- Users can use multiple pipes to display the names of the files in the current directory. In this directory, the file owner can define execute permission by using one single command.

The command is as follows:

```
$ ls -l | grep "^.x" | tr -s " " | cut -d" " -f9
```

The output of the preceding command is as follows:

a.out

test

Here, the output of the **ls -l** command is given as input to the **grep** command.

The **ls -l** command gives a detailed listing of the files in the current directory. The **grep** command extracts all ordinary files that have the execute permission. This output is given to the **tr** command that replaces multiple spaces with a single space.

The **cut** command takes this as input and extracts and displays the ninth field, which is the file name.

- The command to display the file names in the current directory along with the file sizes, one screen at a time is as follows:

```
$ ls -l | tr -s " " | cut -d" " -f9,5
```

The output of the preceding command is as follows:

1384 DEADJOE

1024 Desktop

1024 Mail

12691 a.out

1024 mail

1024 nsmail

58 program.cc

691 test

The command to display the names of all files in the current directory is as follows:

```
$ ls -l | grep "^-" | tr -s " " | cut -d" " -f9
```

The output of the preceding command is as follows:

DEADJOE

a.out

program.cc

test

- The command to display the total number of files in the current directory is as follows:

```
$ ls | wc -l
```

The output of the preceding command is as follows:

16

- The command to display the number of lines where the word ‘and’, occurs in a file ‘test’, is as follows:

```
$ grep -c "and" test
```

The output of the preceding command is as follows:

1

4.7 Check Your Progress

1. Which of the following arguments of the service command updates the configuration file of the service and implements the changes to the service without stopping the service?

(A)	start	(C)	reload
(B)	restart	(D)	status

2. Which of the following utilities configures the current runlevel by default and provides an interactive text-based interface?

(A)	Services configuration	(C)	chkconfig
(B)	ntsysv	(D)	Service command

3. Which of the following runlevels halt the Linux system?

(A)	0	(C)	2
(B)	1	(D)	3

4. Which of the following runlevels start the Linux system, in the full multiuser text mode without enabling a networking service?

(A)	0	(C)	2
(B)	1	(D)	3

5. Which of the following features allow GRUB to boot the operating system from any of the memory addresses, which is either within the first 1,024 cylinders of the disk or beyond?

(A)	Direct and Chain-loading	(C)	Menu interface
(B)	ext2 partition	(D)	LBA

6. Which of the following options of the **date** command displays month of the year in digits?

(A)	%d	(C)	%y
(B)	%m	(D)	%D

7. Which of the following commands clears the standard output device and positions the cursor at the top left corner of the screen?

(A)	clear	(C)	tput clear
(B)	tput cup	(D)	tput rmsso

8. Which of the following commands sets the screen to reverse video?

(A)	clear	(C)	tput clear
(B)	tput cup	(D)	tput rmsso

9. Which of the following commands displays the names of users that are currently logged on to the system?

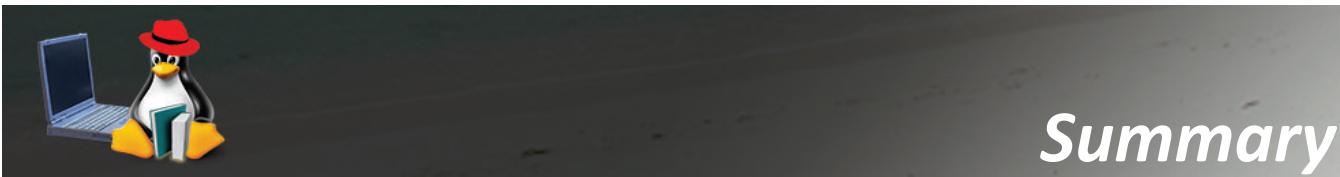
(A)	who	(C)	tput reset
(B)	tput rmsso	(D)	tput cup

10. Which of the following filters is used to search the lines in a file with a particular pattern of characters?

(A)	wc filter	(C)	tr filter
(B)	cut filter	(D)	grep filter

4.7.1 Answers

1.	C
2.	B
3.	A
4.	C
5.	D
6.	B
7.	C
8.	D
9.	A
10.	D



- X Window System is based on the client-server architecture that includes two primary components, X server and X client. The configuration files of the X client and the X server, such as header files, modules, documentation and libraries, are stored in two directories, /etc/x11/ and /usr/x11R6/. The X server and X client can run on same or different systems. The X protocol is a network protocol that is used to establish the client-server relationship between X clients and X servers. To work in the GUI mode on a Linux system, start an X session. To do this, execute the startx command while working in the command-line interface mode.
- Shell is a term used for the interactive-user interface of an operating system. In a multiuser environment, the shell has to isolate the user interface from the kernel. Login shells are interactive shells that set up a certain pre-defined environment when a user logs on. On the other hand, non-login shells do not set any pre-defined environment and they inherit the environment of the parent shell under which they are created.
- A boot loader is a program that loads the operating system into the system memory while booting. Runlevels represent the various modes in which the user can work on a Linux system. Running Linux in different modes helps in utilizing the computing resources efficiently for most of the important tasks at hand.
- RHEL provides some basic commands and command line expansions that help in configuring the system. Some of the basic commands include **date**, **clear**, **tputcup**, and **tput reset**. The command-line expansions include **~,{} and \$**. The **who** command displays the names of users that are presently logged on to the system. In RHEL, users use filters to restructure the output based on a certain condition. Filters are used to work on data in an effective way. Some examples of filters include **grep**, **wc**, **cut**, **tr**, and **sort**.
- RHEL provides a feature called pipe that allows users to combine multiple commands to perform a task. Using pipes, users can direct the output of one command or filter to another command or filter as an input. This way users can combine two commands, which they have to run separately otherwise.

GLOSSARY

look-up guide

Terms Acronyms Abbreviations

ANIMATION **AVIATION** **PROGRAMMING**

Graphics **Html**

Exercise - 1**Displaying the computer name**

Command-line expansion includes the use of tilde (~) command expansion, which includes \$, () or ``. Brace expansion includes { }.

Step 1 - Log on to the RHEL workstation with the username and password.

Step 2 - To open the command-line terminal, click Applications -> System Tools -> Terminal

Step 3 - At the command prompt, type the following command and press ENTER.

```
echo "My computer name is $ (hostname)"
```

Figure 5.1 shows the output of the echo “My computer name is \$ (hostname)” command that displays the computer name.

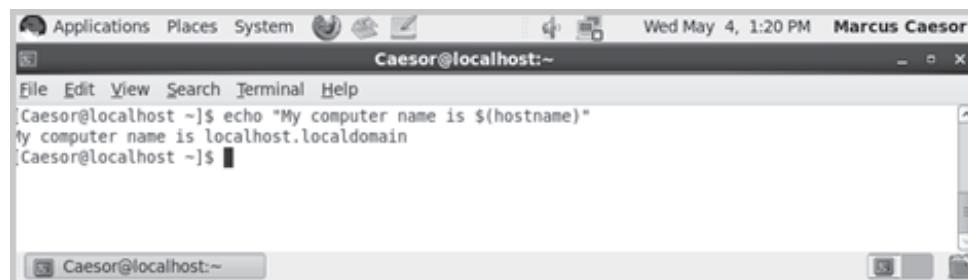

 A screenshot of a Linux desktop environment showing a terminal window. The window title is 'Caesor@localhost:~'. The terminal content shows the command 'echo "My computer name is \$(hostname)"' being run and its output 'My computer name is localhost.localdomain'. The desktop interface includes a menu bar with 'Applications', 'Places', 'System', and icons for system monitoring and file operations. The status bar at the top right shows the date and time as 'Wed May 4, 1:20 PM' and the user as 'Marcus Caesor'.

Figure 5.1: Displaying the Computer Name

Step 4 - Type clear and press ENTER to clear the screen.

Exercise - 2**Creating multiple directories at one time using brace expansion**

Step 1 - At the command prompt, type the following command to display the existing directories and press ENTER.

```
ls
```

Step 2 - Type the following command and press ENTER.

```
mkdir{Aa,Bb,Cc,Dd,Ee}
```

This creates the vrs, Aa, Bb, Cc, Dd, and Ee directories.

Step 3 - Type the following command to verify the folders are created.

```
ls
```

Figure 5.2 shows the created directories using the ls command.

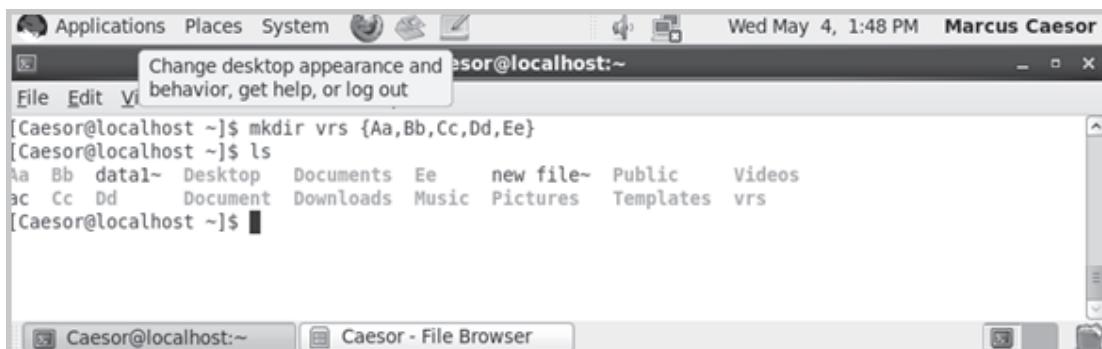


Figure 5.2: Output of the `ls` Command

Step 4 - Type clear and press ENTER to clear the screen.

Exercise - 3

Displaying the home directory

Step 1 - To display the home directory, at the command prompt, type the following command:

```
# echo $HOME
```

Figure 5.3 shows the output of the `echo $HOME` command that displays the home directory.

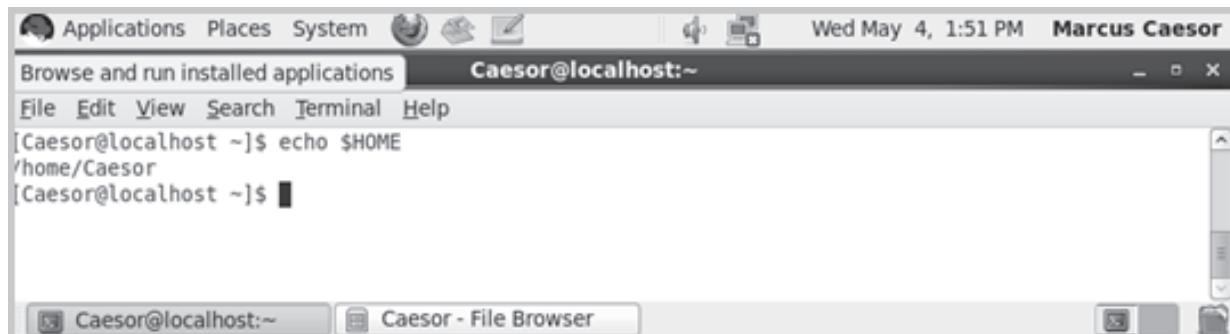


Figure 5.3: Displaying Home Directory

Step 2 - To display the home directory using tilde, type the following command and press ENTER.

```
# echo ~
```

Figure 5.4 shows the output of # echo ~ command.

The screenshot shows a terminal window titled 'Caesor@localhost:~'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The status bar at the top right shows 'Wed May 4, 1:53 PM' and 'Marcus Caesor'. The terminal window contains the following text:

```
[Caesor@localhost ~]$ echo $HOME
/home/Caesor
[Caesor@localhost ~]$ echo ~
/home/Caesor
[Caesor@localhost ~]$
```

Figure 5.4: Output of the echo Command

Step 3 - Type clear and press ENTER to clear the screen.

Exercise - 4

Connecting multiple commands

Step 1 - At the command prompt, type **ls** and press ENTER.

Step 2 - Type **echo \$HOME** and press ENTER.

Note - Both the commands can be combined by separating them with a semicolon.

Step 3 - Type the following command and press ENTER.

```
echo $HOME; ls
```

Figure 5.5 shows the output of the **echo \$HOME; ls** command that combines multiple commands.

The screenshot shows a terminal window titled 'Caesor@localhost:~'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The status bar at the top right shows 'Wed May 4, 2:04 PM' and 'Marcus Caesor'. The terminal window contains the following text:

```
[Caesor@localhost ~]$ echo $HOME; ls
/home/Caesor
.a Bb data1~ Desktop Documents Ee new file~ Public Videos
.ac Cc Dd Document Downloads Music Pictures Templates vrs
[Caesor@localhost ~]$
```

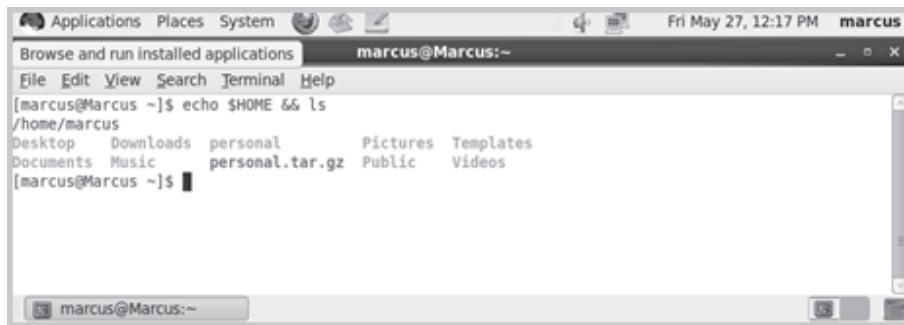
Figure 5.5: Combining Multiple Commands

Note - Multiple commands can also be connected where the second command executes only after the first command is executed.

Step 4 - Type the following command and press ENTER.

```
echo $HOME && ls
```

Figure 5.6 shows the output the echo \$HOME && ls command that connects multiple commands.



The screenshot shows a terminal window titled "marcus@Marcus:~". The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The status bar at the top right shows "Fri May 27, 12:17 PM marcus". The terminal prompt is "[marcus@Marcus ~]\$". The user typed the command "echo \$HOME && ls" and pressed Enter. The output shows the contents of the user's home directory: ".Desktop", "Downloads", "personal", "Pictures", "Templates", "Documents", "Music", "personal.tar.gz", "Public", and "Videos". The terminal prompt "[marcus@Marcus ~]\$" is visible again at the bottom.

Figure 5.6: Connecting Multiple Commands

Step 5 - Type exit to close the terminal.

GROWTH
RESEARCH
OBSERVATION
UPDATES
PARTICIPATION



6.1 Working with File System

A single fixed disk can store thousands of files. RHEL provides a hierarchical file system for grouping files efficiently on the disk. This hierarchical file system is sourced from UNIX. When UNIX was developed, it had features that other operating systems at that time did not have. One such feature was its hierarchical file system.

The file system in UNIX maintains a hierarchy. It allows users to store files under directories. An analogy can be drawn between these directories and the drawers of a filing cabinet. Each drawer of the cabinet contains files, each of these contain documents of similar nature. Similarly, the directories on the disk can be created to store files containing data of a similar nature.

Similar to situation where the user decides to label the drawer and content in a filing cabinet, in RHEL, the user can name the directories and files. RHEL follows the UNIX file system convention.

Figure 6.1 shows a sample Linux file hierarchy structure.

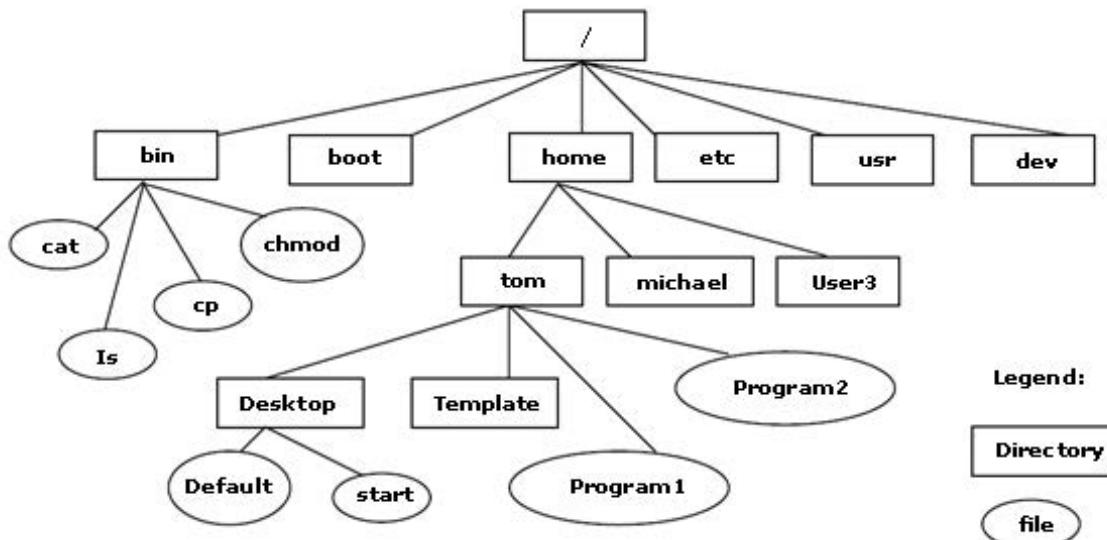


Figure 6.1: Linux File Hierarchy Structure

The files are stored under one main directory, / (root). The directory, /, is further divided into subdirectories such as bin, boot, home, usr, dev, and so on. In each directory, files containing related data are stored. The administrator of the Linux system can place all the home directories of the users under the /home directory. For example, the directory, tom, is the Home directory for the user, tom. He stores all his files in his Home directory tom or creates new directories under it.

The directory tom contains two files, namely Program1 and Program2 and two directories, Desktop and Template. The Desktop directory contains two files, namely Default and start.

A file in the file system hierarchy is referred by its path name. The path name consists of the file name, preceded by the name of the directory where the file is saved. The path name can have a set of directories, one directory containing another, until the root directory is reached. The file name and different directory names contained in the path are separated by the / symbol.

For example, in Figure 6.1, the full path of the file start can be given as /home/tom/Desktop/start.

Similarly, the path for the file ls is /bin/ls. The /symbol is a special character. Therefore, it cannot be included in a file or a directory name.

In a hierarchical, inverted tree-like structure, the operating system provides rapid access to files because the groups of files are isolated from each other. Only one directory (a smaller number of files) has to be searched to locate a file. If there were no subdirectories and all the files were stored under the directory / locating a file would mean searching through the entire list of files until the required file is located.

Some of the directories under the / directory are as follows:

- The /bin directory stores many utilities of Linux. These utilities are the commands available under the Linux system. They are in the binary format, hence the name 'bin'.
- The /dev directory stores all the device-related files for the system.
- The /etc directory stores system-related data that has to be referred by the users and the system. For example, the passwd file. This data can include some important programs or files containing configurations for different applications.
- The /lib directory stores libraries of data for the compilers installed on the system, such as the C language routines.
- The /home directory contains all the Home directories of users.
- The /usr directory stores the operating systems files, that are not involved in the boot process. Many utilities are available in the bin directory under the usr directory. The /usr/bin directory is different from the /bin directory.
- The /var directory stores the information related to different utilities of Linux. Each of these directories is organized to store a specific type of file.

6.1.1 Types of Files

In Red Hat Enterprise Linux, the information is considered as a file. Therefore, apart from a user's program files and data files, there are special files, such as files that contain information about directory content or the various I/O devices connected to the system.

A device is also treated as a file and all the information being transferred to the Visual Display Unit is treated as a file.

The different types of files in RHEL are as follows:

- **Ordinary Files** - These files are created by a user. These include all the data files, program files and executable files. These types of files can be part of the operating system or a user can create them.

- **Directory Files** - These files are created automatically when a user executes the `mkdir` command. These files are saved with the same name as the directory. The file stores the directory related information, such as name, size, and timestamp. Information about any content that the user adds to the directory later is also stored in this file. For example, for the directory `/home/steve`, there is a directory file `steve` in the directory, `/home`. The information about all the files and directories under the directory, `steve` is stored in this directory. A user cannot modify a directory file. When a new file or a subdirectory is added to a directory, it is modified automatically.
 - **Special Files** - These are system files used by RHEL that are typically associated with I/O devices. These files are stored in directories, such as `/dev` and `/etc`. Special files cannot be modified by the users. The different types of special files, supported by RHEL are as follows:
 - **Character Device Files** - These files perform read and write operation one character at a time. An example of a character device file is `modem`.
 - **Block Device Files** - These files can access a block of data at a time. A block of data can hold either 512 bytes or 1,024 bytes. A minimum of one block of data at a time is read or written by the kernel. The data is collected in the memory for the block memory by the kernel. It is then made available to a user. Such random access makes I/O operations quick in devices. An example of a block device is the hard disk. Many of the devices can act as character devices or block devices depending on the command that is used to access the device.
 - **Hard Links** - These files allow one file to have multiple names. Only a file can have a hard link file while directories cannot have hard link files. Hard link files create a direct link to a data structure. As each file system has its own data structure information, users can hard-link files only when they are on the same file system. Symbolic links do not have this restriction.
 - **Symbolic Links** - These files are similar to hard links. However, the symbolic link files can work across different file systems. These are also called soft links.
 - **Sockets** - These are interfaces between processes running on different computers that help in communication. Sockets work under the protection of file system access control.
 - **Named Pipes** - Transfer data between two processes running on the same computer where the output of one or more processes acts an input to another process. Hence, named pipes are also referred to as First-In First-Out (FIFO) files.
 - **Archiving Files** - When users store a group of files in one file, it is called archiving. For example, users have to backup the file, transfer the files to another directory or a different computer or save the disk space. In such situations, the user can choose to archive or compress the files. To archive the files, the user can remove the source data and keep the archived data after the archiving is completed. The collection of files and directories stored in one file is called archiving files. These files occupy the same disk space. This is because all the individual files and directories are just combined together.
 - **Compressed Files** - If a user does not require discarding the source data, the user compress it.
-

Compressed file data storage reduces the size of the files as compared to their actual sizes. For example, a user can use a compressed file if there is not enough disk space on the computer to keep data in its actual size.

Note - Users can also choose to create an archive file and then compress it to save disk space.

6.2 Managing Files and Directories at the Command Line

There are various commands used in RHEL to manage directories and files at the command line, such as the **cat**, **cd**, **ls**, **mkdir**, **more**, **less** and **file**. In addition, there are various text editors to manage files. A text editor is used to create and manage text files and documents and has a wide variety of features, such as creating a file, copying and pasting text and searching text. RHEL has large number of available editors, many of them inherited from UNIX, such as **emacs**, **vi**, **sed**, **gedit** and **vim**.

6.3 Manipulating Files

File is a collection of information stored as a single unit with a unique filename. There are a number of commands readily available in RHEL to work with files. These commands help users to create, modify, remove, move, or copy files.

Some of the commands listed are as follows:

- **touch**
- **cp**
- **rm**
- **mv**

Users can work with a set of files without providing the names of each file by using a wildcard character. It uses special characters that match a pattern to specify a file or a set of files.

Creating a File

Users can create a new file using **vi** or **vim** editors. To create a new file using the **vi** editor, users can execute the following command:

```
vi testfile.txt
```

This command opens the **vi** editor. Users can then add content to the file using the **vi** commands. After users save the changes, a new file **testfile.txt** is created with the changes made by them.

In the preceding example, the **vi** editor shows the contents of the file if it already exists.

Alternatively, users can use the **touch** command to create a blank file. The syntax of the **touch** command is as follows:

```
touch [options] [filenames]
```

If users execute the **touch** command on an existing file, only the **timestamp** of the file changes to the current time.

Example:

Users can execute the following command to change the **timestamp** of the **testfile.txt** file:

```
touch testfile.txt
```

In the preceding example, the **timestamp** of the **testfile.txt** is changed if the file already exists. However, if the file does not exist, touch creates a new file with the name **testfile.txt**. In case users do not require a new file to be created while using the **touch** command, they can use the **-c** option as shown in the following example:

```
touch -c testfile.txt
```

If users use the **touch** command to change the **timestamp** of a file, the access and modification time of the file is changed. However, users can only change the access or modification time of the file using one of the following options:

- ➔ **-a** - changes the access time to the current time
- ➔ **-m** - changes the modification time to the current time

Users can also use any graphical text editor like **gedit** to create a file in the GUI mode.

Copying Files

The **cp** command duplicates the content of the source file into a target file. The syntax of the **cp** command is as follows:

```
cp [options] <source file/s><destination directory/file>
```

where,

the argument, **<source file/s>**, specifies a single file or multiple files to copy

the argument, **<destination directory>**, specifies the location where users have to copy the file.

the argument, **<destination file>**, specifies the name of the copy of the file,

<source file/s>. Users can specify either of the two arguments, **<destination directory>** or **<destination file>**.

The following command copies the content of the file ‘data1’ to a new file ‘data2’:

```
cp data1 data2
```

In the preceding example, if the file ‘data2’, already exists, its content is overwritten with the content of the file ‘data1’. Users can specify the complete path with the **cp** command to copy files across directories.

Users can also copy a directory recursively using the **cp** command with the -r option.

```
cp -r temp tempo
```

This command copies the temp directory and all its files and subdirectories to the tempo directory. If the directory, tempo, already exists, all the content is copied within that directory. Otherwise, the directory, tempo, is created in the current working directory.

Note - When users copy files using the **cp** command, it does not preserve Access Control Lists (ACLs).

Table 6.1 lists the other common options used with the **cp** command.

Option	Function
-i	Prompts before overwriting a file or directory
-l	Creates a link to the file instead of copying it
-s	Creates a symbolic link
-v	Explains the process in detail

Table 6.1: Options with the **cp** Command

Removing Files

The **rm** command is used to delete files or directories. The syntax of the **rm** command is as follows:

```
rm [options] file/s
```

where,

the argument, file/s specifies the name of one or more files to be deleted.

Example:

To remove the files, data1 and data2, from the current directory, users can use the following command:

```
rm data1 data2
```

If the file to be deleted is not located in the current directory, users have to provide the complete path along with the file name. The following command removes the file, data1, that is stored in the directory, /home/steve:

```
rm /home/steve/data1
```

The -r option is used with the **rm** command to remove a directory along with its subdirectories. This option is sometimes preferred over the **rmdir** command because in the case of **rmdir**, users can only delete an empty directory.

For example, to remove the tempo directory along with all its subdirectories, users can use the following command:

```
rm -r tempo
```

Table 6.2 lists the commonly used options with the **rm** command.

Option	Function
-i	Prompts before removing a file or directory.
-f	Removes a file by force. It ignores the nonexistence of a file. The command does not flag an error if the file does not exist.
-r or -R	Deletes recursively. It deletes a directory along with its subdirectories.
-v	Explains the process in detail.

Table 6.2: Options with the **rm** Command

Moving and Renaming Files

The **mv** command is used to move a file or a directory from one location to another or to change the name of a file or a directory. Moving a file from one location to another is different from copying a file. While moving a file, no new file is created.

The syntax of the **mv** command is as follows:

```
mv [option] <source> <destination>
```

where,

the argument, source specifies the directory or file that users require to move.

The argument, destination specifies the location where the file or directory is to be moved.

Example:

The following command shows how a file can be moved to another directory:

```
mv data3 /home/steve/programs/data3
```

In the preceding example, the file, 'data3' is moved from the current directory to the /home/steve/programs directory. Users can also use the **mv** command to rename a directory, as shown in the following command:

```
mv comm communication
```

In the preceding example, the directory, comm, is renamed to communication. This is because the directory, 'communication' does not exist in the current directory.

In addition to renaming a directory, users can use the **mv** command to move a directory to a new location, as shown in the following command:

```
mv communication temp
```

In the preceding example, the directory, temp, already exists. Therefore, the communication directory is

moved from the current directory to the temp directory.

Table 6.3 lists some of the options available with the `mv` command.

Option	Function
<code>-f</code>	Does not prompt before overwriting a file or directory
<code>-i</code>	Prompts before overwriting at the destination location
<code>-v</code>	Explains the process in detail

Table 6.3: Options with the `mv` Command

Inserting and Manipulating Text

Users can insert or add text in the existing file using the `vi` editor. Table 6.4 lists the commands used to insert, add, and replace text in a file.

Option	Function
<code>a</code>	Appends text after the current cursor position
<code>A</code>	Appends text at end of the current line
<code>i</code>	Inserts text before the current cursor position
<code>I</code>	Inserts text at beginning of the current line
<code>o</code>	Inserts a blank line below the current line and allows to append text
<code>O</code>	Inserts a blank line above the current line and allows to append text
<code>rx</code>	Replaces the current character with character x
<code>Rtext</code>	Replaces characters with the specified text (until Esc key is pressed). This command activates the replace mode instead of the append mode.

Table 6.4: Insert, Add, and Replace Commands

Note - To use these commands in the `vi` editor, users must be in the Insert mode. Therefore, press `ESC` to terminate the text entry and return to the Command mode.

Changing and Yanking/Cutting the Text

In RHEL, there are different commands that help users to change, yank and paste the yanked text.

Table 6.5 lists the commands to change and yank the text.

Option	Function
<i>cw</i>	Changes a word
<i>cc</i>	Changes a complete line
<i>j</i>	Joins lines
<i>U</i>	Restores the last change
<i>X</i>	Deletes a character before the current cursor position
<i>u</i>	Undoes the last change
.(dot)	Repeats the last change
<i>Yy</i>	Copies and yanks/cuts the current line into the buffer
<i>Nyy or yNy</i>	Copies and yanks/cuts the current and the other lines into the buffer
<i>P</i>	Places the yanked text after the current cursor position. This command is similar to the paste operation in Windows.

Table 6.5: Commands to Change and Yank the Text

6.4 Commands Used in *vi* Editor

There are various types of commands used in *vi* editor that are as follows:

- Commonly used commands
- Cursor movement commands
- Insert and replace commands
- Delete and modify commands
- Commands to copy and paste lines

Table 6.6 displays the commonly used commands in *vi*.

Command	Action
H	Moves cursor to previous character
L	Moves cursor to next character
K	Moves cursor up one line
J	Moves cursor down one line
X	Deletes character at current cursor
dd	Deletes line
:e!	Undoes unsaved work

Command	Action
:wq+ENTER	Saves all changes and quit
:W+ENTER	Saves the file
:q!+ENTER	Quits without saving changes
:e <filename>+ENTER	Opens the specified file
:w <filename>+ENTER	Writes to a different file
:w! <filename>+ENTER	Forces write to another file
:! <commandname>+ENTER	Executes a shell command

Table 6.6: Commonly Used Commands

Table 6.7 displays the basic cursor movement commands in vi.

Command	Action
Ctrl+d or D	Moves to the last line on the screen
Ctrl+u or U	Scrolls up half screen
Ctrl+F	Moves one page forward
Ctrl+B	Moves one page backward
nG	Moves to line number n
0 (zero)	Moves to the beginning of the line
\$	Moves to the end of the line
H	Moves to the first line on the screen
M	Moves to the middle line on the screen
L	Moves to the last line on the screen
Z+Enter	Makes the current line the first line on the screen
z-	Makes the current line the last line on the screen

Table 6.7: Cursor Movement Commands

Table 6.8 displays the basic insert and replace commands in vi.

Command	Action
a	Appends after current character
A	Appends at the end of line
i	Inserts before the current character
I	Inserts at the beginning of line
o	Inserts blank line below and allows insertion
O	Inserts blank line above and allows insertion
Rx	Replaces current character with character x

Command	Action
H	Moves to the first line on the screen
M	Moves to the middle line on the screen
L	Moves to the last line on the screen
Z+Enter	Makes the current line the first line on the screen
z-	Makes the current line the last line on the screen

Table 6.8: Insert and Replace Commands

Table 6.9 displays the basic deletion and modification commands.

Command	Action
dw	Deletes word
dd	Deletes line
cw	Changes word
cc	Changes line
X	Deletes character before current cursor position
I	Joins lines
u	Undoes last change
U	Restores last change
. (dot)	Repeats last change

Table 6.9: Deletion and Modification Commands

Table 6.10 displays the basic commands to copy and paste lines.

Command	Action
yy	Copies the current line
nyy	Copies n lines from the current line
P	Places the copied line after current cursor position
P	Places the copied line before the current cursor position

Table 6.10: Copy and Paste Lines Commands

6.4.1 Getting Started with gedit Editor

The gedit editor is a text editor. It is used in the Gnome desktop environment. To open gedit, go to Applications → Accessories → gedit text editor.

Figure 6.2 displays the Readme tab of gedit editor.

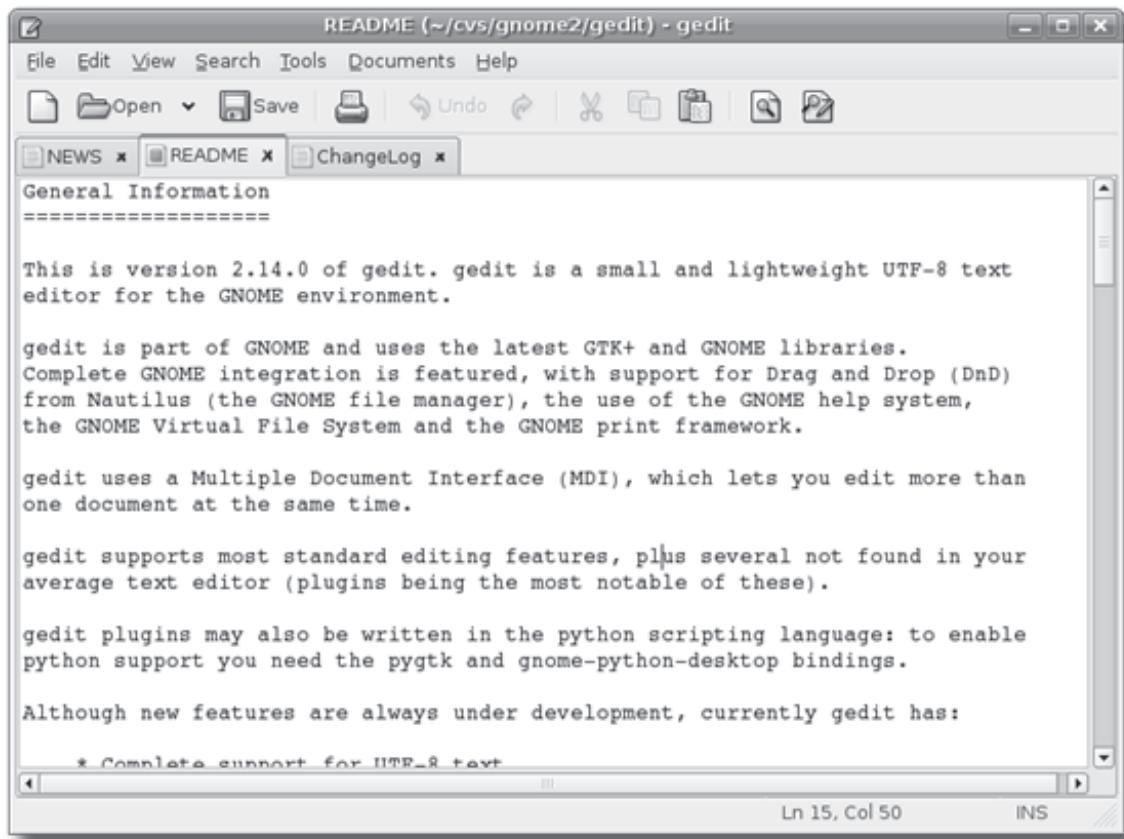


Figure 6.2: Readme Tab of gedit Editor

The gedit editor provides a variety of features that are as follows:

- Options to highlight configurable syntax for various languages, such as C, C++, Java, HTML, XML and Perl
- Options to Undo/Redo
- Access to editing files from remote locations
- Options on file reverting
- Support for print and print preview options
- Support for clipboard options, such as cut, copy and paste
- Options to search and replace text

- Options to go to a specific line
- Option for auto indentation
- Option for wrapping text
- Option to display line numbers
- Option to highlight current line
- Option to backup files
- Configure fonts and colors

6.5 Managing Files with Nautilus

Nautilus is a multipurpose tool that is included in RHEL. There are multiple file management activities, such as file management and managing remote computers that can be performed using this tool.

Nautilus provides a graphical user interface similar to Windows Explorer to manage files. Using Nautilus, users can drag and drop, copy, move or perform other regular operations on files without executing any commands.

Figure 6.3 shows a typical file browser window provided by the Nautilus.

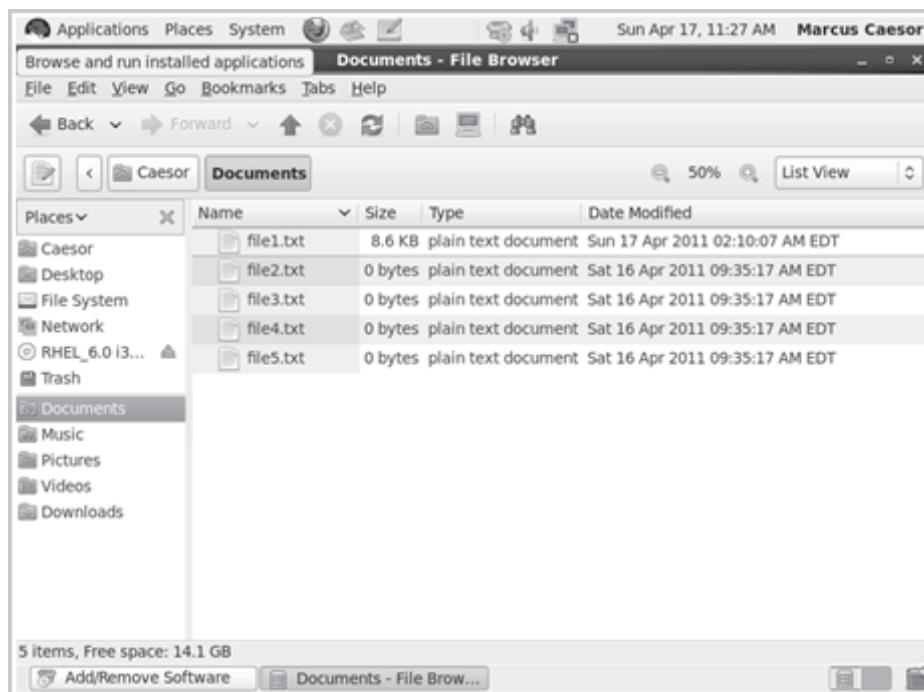


Figure 6.3: Nautilus - File Browser Window

6.5.1 Redirecting I/O Channels to Files

Redirection changes the assignments for the standard input, output and error. Using redirection, the input to a command can be taken from a file other than the keyboard. Similarly, the output of a command or the error can be written to a disk file or printed, instead of displaying it on the screen. The three types of redirection are as follows:

- Input Redirection
- Output Redirection
- Error Redirection

Input Redirection

The following example illustrates the use of input redirection:

```
cat< test1
```

Here, the less than symbol (<), implies input redirection from the file, **test1**. The **cat** command takes each line of the file ‘**test1**’ as input and displays it on the screen. Therefore, it is possible to specify that the standard input comes from a disk file. The preceding command can also be written using the file descriptor that is as follows:

```
$ cat 0< test1
```

Here, ‘0’ indicates input redirection.

This can be useful in scenarios where users do not require the input for any command to be entered with a keyboard. A typical example can be a shell script where the input for a command comes from a file rather than, a keyboard.

Output Redirection

The following example illustrates the usage of output redirection:

```
cat test1 > out_test
```

Here, the greater than symbol (>) implies redirection of output to the file, **out_test**. The output of the **cat** command is written to a disk file ‘**out_test**’ instead of displaying it on the screen. In output redirection, the file to which the output is redirected is first created on the disk as an empty file and then the output is sent to this file. However, if the file already exists, its contents are deleted before the output is written to it.

The following command appends the output to the existing file, **out_test**:

```
cat test1 >> out_test
```

The preceding commands can also be written using the file descriptor as:

```
cat test1 1> out_test
```

```
cat test1 1>>out_test
```

Here, 1 indicates output redirection. Although the file descriptor '1' is not necessary, it can be included for clarity.

Error Redirection

The following example illustrates the usage of error redirection:

```
cat test_2 2> error-msg
```

The file 'test_2' does not exist in the current directory. Therefore, when a user tries to execute this command, Linux generates an error message because the execution is unsuccessful. This error message is written to the file, error-msg, instead of displaying it on the screen (the standard error file). As in the case of output redirection, error redirection also first creates the file to which the error messages are redirected and then writes the error output to the file.

6.6 Handling Files Using Wildcard Characters with File Operations

The shell offers the feature to perform an operation on a set of files without specifying the names of all the files on which the operation is to be performed. This is made possible by the use of certain special characters in the command in place of the actual filenames. The shell interprets these special characters as a specific pattern of characters. The shell then compares all the file names under the directory specified in the command to find the file names that match the pattern. The command is executed on the files with names that match the pattern.

Table 6.11 lists the various wildcard characters available.

Option	Description
*	Matches none or one character or a string of characters
?	Matches exactly one character
[]	Matches exactly one of a specified set of characters

Table 6.11: Wildcard Characters

The * Wildcard

The * wildcard is interpreted as a string of none, one or more characters. The following example demonstrates the usage of * wildcard with the **cat** command:

```
# cat c*
```

Here, c* matches the file names that start with the character c. The * wildcard can also be repeated in the command-line as shown in the following example:

```
# cat chap*.*
```

This command displays all the files that start with 'chap' and containing any sequence of characters or no character followed by any file extension.

The ? Wildcard

The ? wildcard matches exactly one occurrence of any character. The following example demonstrates the usage of ? wildcard:

```
ls *?
```

This command displays all the files with one or more characters as the filename and a single character as the extension.

The [] Wildcard

The [] wildcard can be used to restrict the characters to be matched.

The following example demonstrates the usage of the [] wildcard:

```
cat a[123]
```

This command displays the content of the files with two character file names that start with the character 'a' and with the next character as 1, 2 or 3, such as a1, a2 or a3.

6.6.1 Searching Files

RHEL allow users to search documents. The following commands helps users to search their documents:

→ *find Command*

→ *locate Command*

The **find** Command

Users can find files in RHEL using the **find** command. The **find** command allows users to use various parameters to make searches efficient and accurate. Using the **find** command, users can locate files on a RHEL system. Users can use different parameters to filter and locate files.

The syntax of the **find** command is as follows:

```
find<location><options><file_name>
```

where,

<location> - specifies the place where users can search the file.

<options> - specifies the options that users can use with the **find** command.

<file_name> - specifies the name of the file users have to search.

For example, the following command searches for a file named **test.txt** and the search starts from the root directory/and traverses through all the subdirectories recursively:

```
find / -name 'test.txt'
```

In the preceding command, '/' specifies the search to start from the root directory. If users do not specify

any location, the search starts from the current directory and goes on recursively. The **-name** option specifies the name of the file to be searched. Users can also use the **-iname** option to make the search case insensitive.

Users can also use the **find** command to search files based on its size.

For example, the following command searches for all the files that are greater than 500 KB in size in the root directory:

```
find / -size +500k
```

To check for files less than a certain size limit, users can use **-** sign while specifying the size limit with the **-size** option. For example, users can use **-size -500k** option in the previous command to search for files less than 500 KB in size in the root directory.

Users can also search files based on the last accessed time using the **find** command. For example, the following command searches for all the files accessed in the last 10 minutes that start with the character **a**:

```
find / -amin -10 -name a*
```

Users can also search files based on the last modified time using the following command:

```
find / -mmin -10 -name a*
```

In this case, only the files that were modified in the last 10 minutes are displayed. Users can also use the **-or** and **-and** options with the **find** command to check for any conditions during a search. For example, the following command searches for all the files that start with the character **'a'** and are more than **500 KB** in size:

```
find / -name a* -and -size +500k
```

The following command searches for all the files that start with the character **'a'** or are less than 500 KB in size:

```
find / -name a* -or -size -500k
```

The following command searches for files that do not start with the character **'a'** and are greater than 500 KB in size:

```
find / -size +500k ! -name a*
```

The locate Command

RHEL allow users to search files using the **locate** command. Users can use the **locate** command to find any files on which they have read permissions. The **/** command maintains a list of files on the system in a database called **/var/lib/mlocate/mlocate.db**.

The syntax of the **locate** command is as follows:

```
locate<file_name>
```

The **locate** command result returns all the files with the **<file_name>** pattern in their names along with

their complete path. If users have to search for a file with exact name, they can use the `-b` option.

The following command returns files with the `<file_name>` as their name:

```
locate -b <file_name>
```

The **mlocate** database used by the **locate** command is maintained by a command **updatedb**. This command is executed once in a day by default. Therefore, users should use **locate** command for files that are at least a day old.

If users have to search for files created immediately or files that are less than a day old, users can execute the **updatedb** command manually before executing the **locate** command to search for files. To execute the **updatedb** command, the syntax is as follows:

```
updatedb
```

However, users must have root privileges to execute the **updatedb** command.

Alternatively, users can also use the GNOME search tool to search files in GUI mode. This tool provides a graphical interface to search and locate files.

Users can invoke this tool using the following command:

```
gnome-search-tool&
```

6.7 Check Your Progress

1. Which of the following directories store utilities of Linux?

(A) /bin	(C) /etc
(B) /dev	(D) /lib

2. Which of the following directories store the operating system files that are not involved in the boot process?

(A) /bin	(C) /etc
(B) /dev	(D) /usr

3. Which of the following types of files are created by a user that includes all the data files, program files and executable files?

(A) Ordinary files	(C) Special files
(B) Directory files	(D) Character device files

4. Which of the following commands are used to display the content of the specified file one screen at a time?

(A) cat	(C) more
(B) less	(D) vi

5. Which of the following commands in vi editor are used to move the cursor to the previous character?

(A) L	(C) K
(B) H	(D) J

6. Which of the following commands in vi editor are used to move the cursor to the previous character?

(A) cat	(C) locate
(B) touch	(D) find

7. Which of the following wildcard characters match exactly one of the specified set of characters?

(A)	*	(C)	!
(B)	[]	(D)	?

8. Which of the following commands delete files recursively?

(A)	<i>rm -R</i>	(C)	<i>rm -i</i>
(B)	<i>rm -f</i>	(D)	<i>rm -d</i>

9. Which of the following commands consult a database before searching for a file?

(A)	<i>find</i>	(C)	<i>mv</i>
(B)	<i>cat</i>	(D)	<i>locate</i>

10. Which of the following commands rename a file or directory?

(A)	<i>cp</i>	(C)	<i>mv</i>
(B)	<i>rm</i>	(D)	<i>cat</i>

6.7.1 Answers

1.	A
2.	D
3.	A
4.	C
5.	B
6.	D
7.	B
8.	A
9.	D
10.	C



Summary

- All the files are stored on the disk under one main directory, / (root). The different types of files are ordinary files, directory files and special files. There are various commands used in Red Hat Linux to manage files at the command line, such as cat, more, less, and file. There are also various editors to manage files, such as vi and gedit.

Are you looking for online **HELP?**

We are just a *click* away



To chat with a

login to **www.onlinevarsity.com**

Session - 7

File System Basics (LAB)

Welcome to the Session, **File System Basics (LAB)**.

This session provides practice on checking and modifying system date and time using command line interface. Further, this session explains how to compress and decompress files, how to organize the files and directories. Lastly, session explains how to various file operations and how to create symbolic and hard links.

In this Session, you will learn to:

- Change date and time manually using command line
- Perform compression and decompression of files
- Perform file operations
- Create symbolic and hard links



Exercise - 1**Changing the date and time manually**

After the user logs on to the system, the user can change the date and time from the menu bar.

Step 1 - Click **System → Administration → Date and Time**. Alternatively, the user can enter **systemconfig-date** at the command prompt.

Note - As user typically is not logged on as root user, the user is prompted for the root password to make necessary changes.

Step 2 - After the user provides the root user password, the Date/Time Properties screen appears as shown in Figure 7.1. Set the date and time as required.

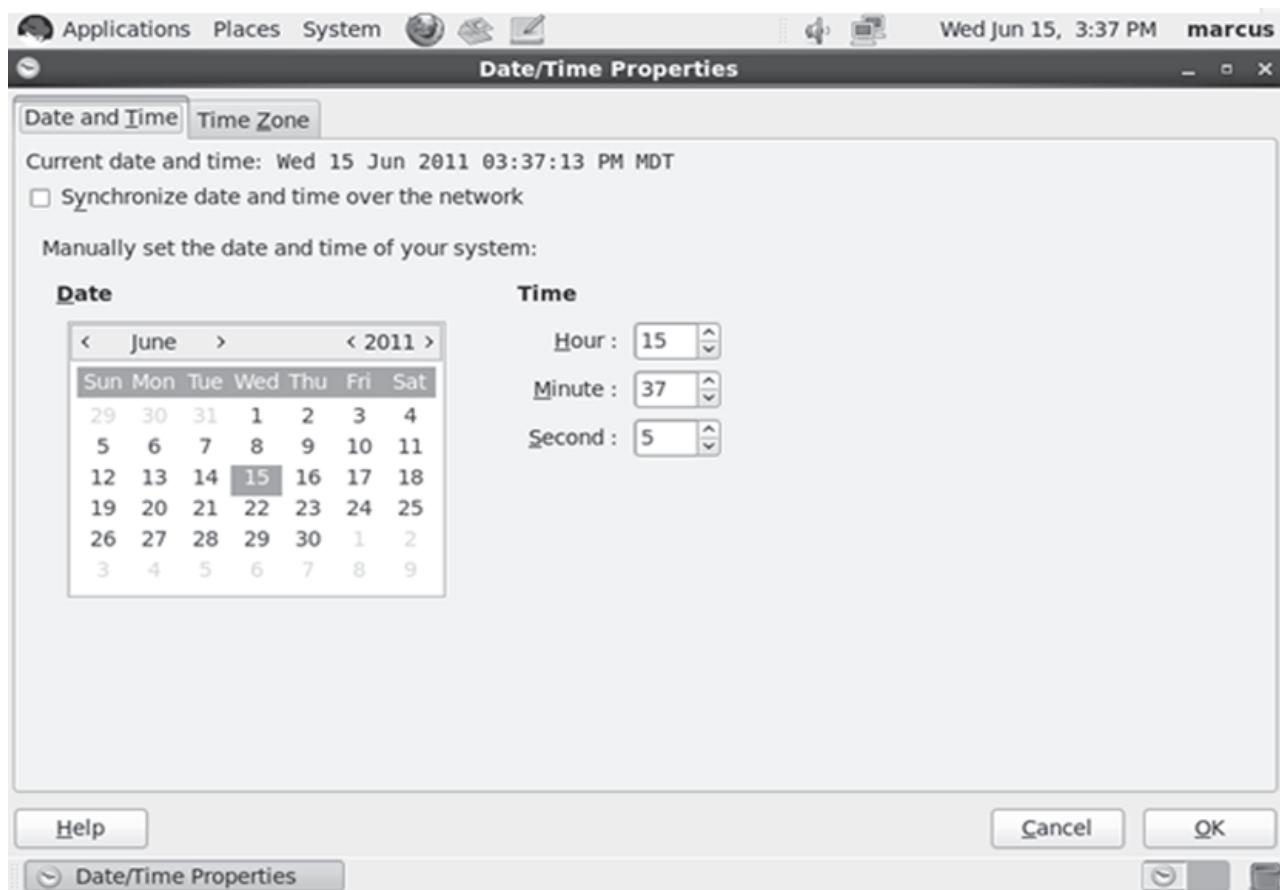


Figure 7.1: Date/Time Properties Screen

Step 3 - Click the Time Zone tab as shown in Figure 7.2. The user can change the time zone settings from this tab. The system clock time is adjusted automatically based on the time zone selected. Click **OK**.

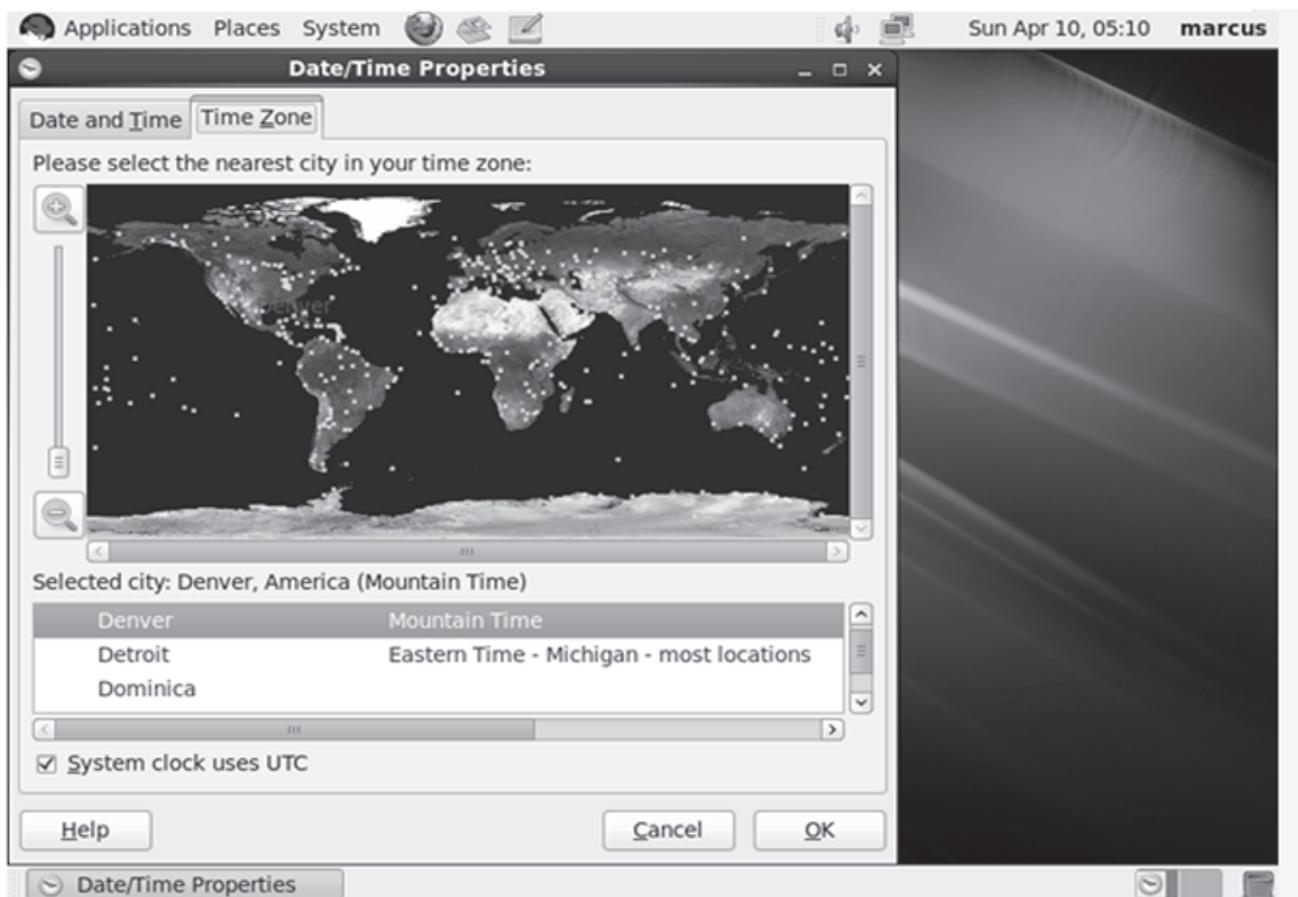


Figure 7.2: Time Zone Tab

This method is the manual way of configuring time for a computer that is running RHEL. If there are a large number of computers that exists on the network, the user can choose to synchronize all the computers with a time server that either exists on internal network or outside the network.

RHEL allows a user to add local or remote time servers. By default, there are three time servers that are added when the Synchronize date and time over the network option is selected as shown in Figure 7.3.

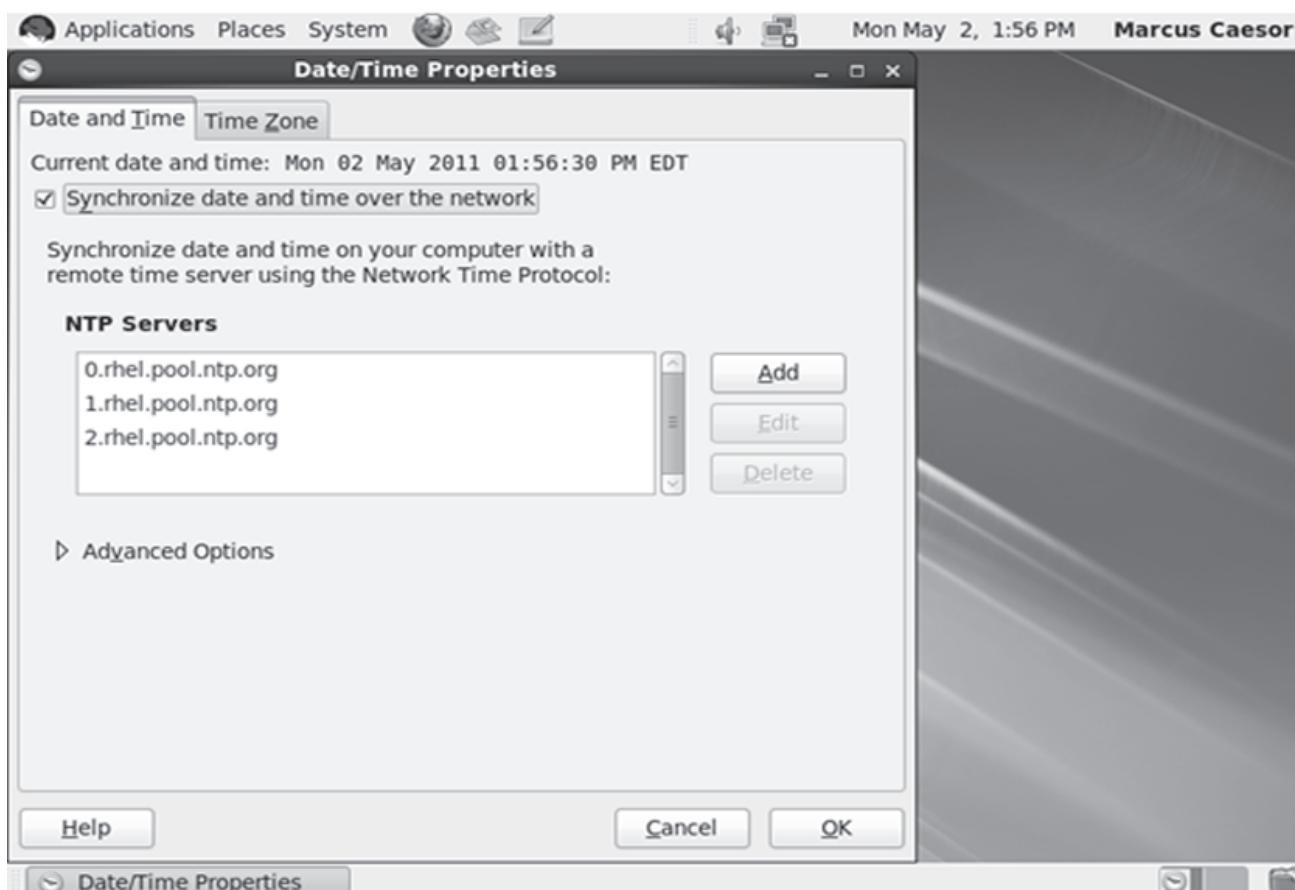


Figure 7.3: Selecting Synchronize date and time over the network Option

Exercise - 2

Changing the date and time using command line

Step 1 - Click Applications → System Tools → Terminal to open the terminal.

Step 2 - Execute the `su` command and log on as root user.

Step 3 - Verify the current date of the system by executing the `date` command:

```
date
```

The output of the **date** command is shown in Figure 7.4.



A screenshot of a Linux desktop environment showing a terminal window. The window title is "root@localhost:~". The terminal shows the following session:

```
[Caesor@localhost ~]$ su - root
Password:
[root@localhost ~]# date
Sat Apr 30 08:31:05 EDT 2011
[root@localhost ~]#
```

Figure 7.4: Output of the **date** Command

Step 4 - To change the date and time, execute the following command:

```
date -s "1 MAY 2011 18:00:00"
```

Step 5 - Execute the **date** command and press ENTER.

The result will display the newly set date and time, as shown in Figure 7.5.

A screenshot of a Linux desktop environment showing a terminal window. The window title is "root@localhost:~". The terminal shows the following command sequence:

```
[Caesor@localhost ~]$ su - root
Password:
[root@localhost ~]# date
Sat Apr 30 08:31:05 EDT 2011
[root@localhost ~]# date -s "1 MAY 2011 18:00:00"
Sun May 1 18:00:00 EDT 2011
[root@localhost ~]#
```

Figure 7.5: Newly Set Date and Time

To add more time servers, click **Add** and specify the name of the time server. It is always advisable to add a local time server to the computers that are running on an organization's network. The key benefit is that all computers receive the same time from the internal time server.

The internal time server that a user configures also acts as a client to another time server. To configure a server to act as a time server, the user must modify the **/etc/ntp.conf** file.

Note - A time server receives and sends information using User Datagram Protocol (UDP) connections on port 123.

Exercise - 3**Compressing and decompressing files with tar and gzip.**

In this exercise, a number of text files that exist in the Documents directory must be compressed. If the text files do not exist, first create five text files named **file1.txt**, **file2.txt**, **file3.txt**, **file4.txt**, and **file5.txt** respectively. Then, create a **tar.gz** file in the Documents directory, and then decompress these files in the same directory. Ensure that the individual files are deleted before decompressing.

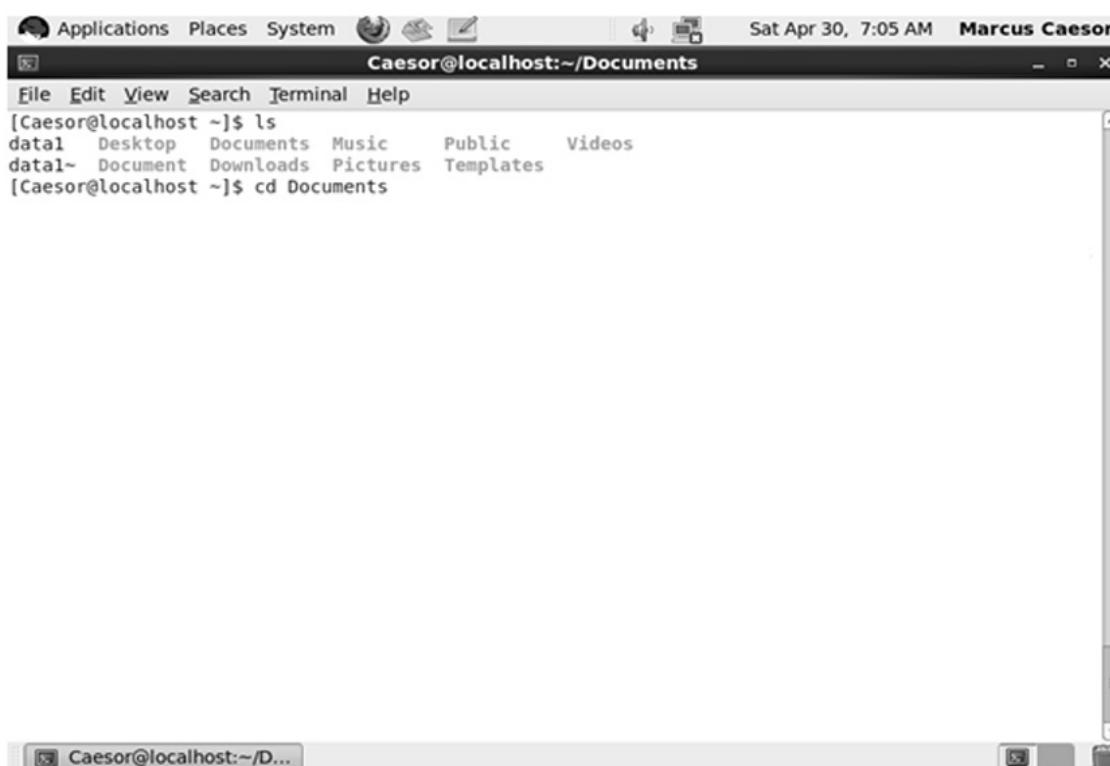
Step 1 - Log on to the RHEL system with the login credentials.

Step 2 - Click Applications → System Tools → Terminal to open the terminal.

Step 3 - At the command prompt, navigate to the Documents directory using the following command:

```
# cd Documents
```

The /Documents directory is displayed in Figure 7.6.



A screenshot of a terminal window titled "Caesor@localhost:~/Documents". The window shows the following command-line session:

```
[Caesor@localhost ~]$ ls
data1 Desktop Documents Music Public Videos
data1~ Document Downloads Pictures Templates
[Caesor@localhost ~]$ cd Documents
```

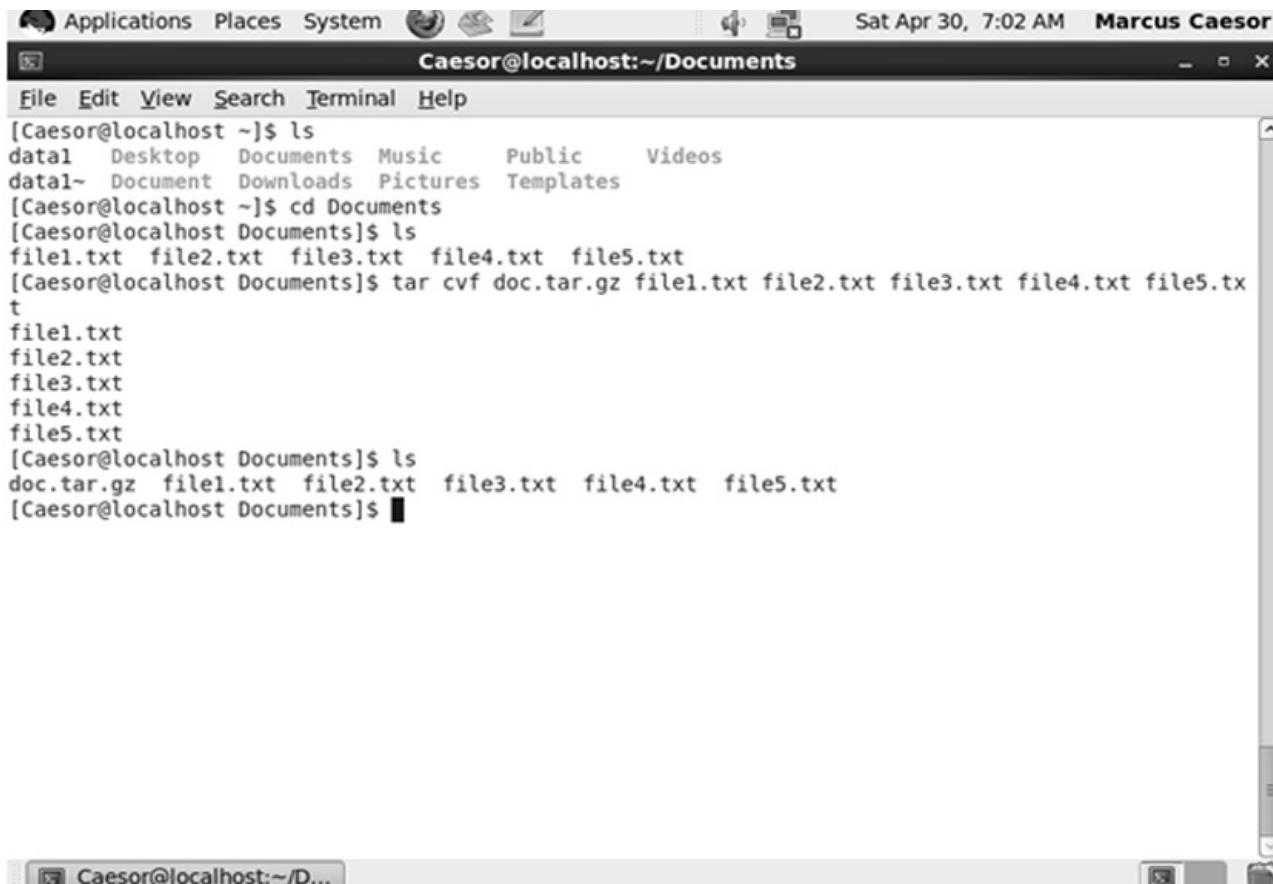
Figure 7.6: Documents Directory

Step 4 - Execute the following command to compress the files with the tar and gzip utilities:

```
# tar cvf doc.tar.gz file1.txt file2.txt file3.txt file4.txt file5.txt
```

The doc.tar.gz is created in the /Documents directory.

Step 5 - Execute the ls command to verify the file existence in the /Documents directory as shown in Figure 7.7.



A screenshot of a terminal window titled "Caesor@localhost:~/Documents". The window shows the following session:

```
[Caesor@localhost ~]$ ls
data1 Desktop Documents Music Public Videos
data1~ Document Downloads Pictures Templates
[Caesor@localhost ~]$ cd Documents
[Caesor@localhost Documents]$ ls
file1.txt file2.txt file3.txt file4.txt file5.txt
[Caesor@localhost Documents]$ tar cvf doc.tar.gz file1.txt file2.txt file3.txt file4.txt file5.txt
[Caesor@localhost Documents]$ ls
doc.tar.gz file1.txt file2.txt file3.txt file4.txt file5.txt
[Caesor@localhost Documents]$
```

Figure 7.7: Executing the ls Command

Step 6 - Delete the individual files by executing the following command:

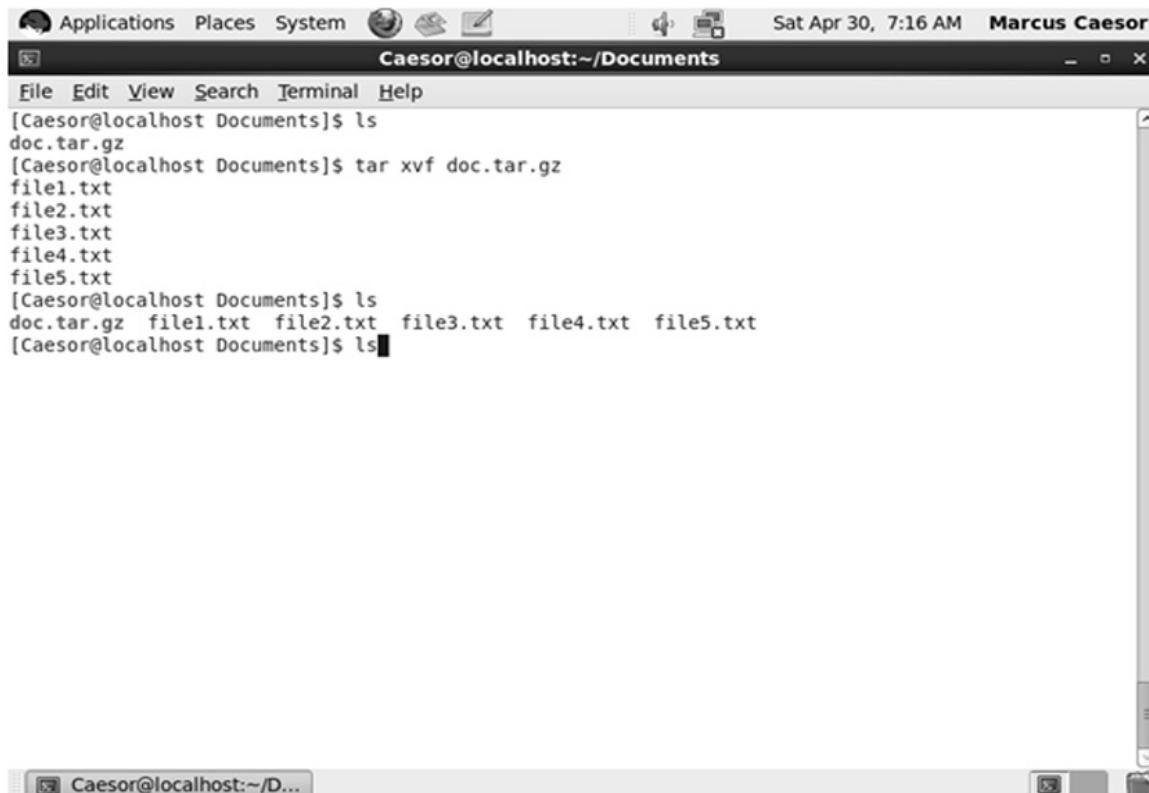
```
# rm file1.txt file2.txt file3.txt file4.txt file5.txt
```

All text files from the /Documents directory are deleted. Only the .tar.gz file would remain in the directory.

Step 7 - Decompress the .tar.gz file in the same directory, by executing the following command:

```
# tarxvf doc.tar.gz
```

The files are extracted in the same directory as shown in Figure 7.8.



A screenshot of a terminal window titled "Caesor@localhost:~/Documents". The window shows the following command sequence:

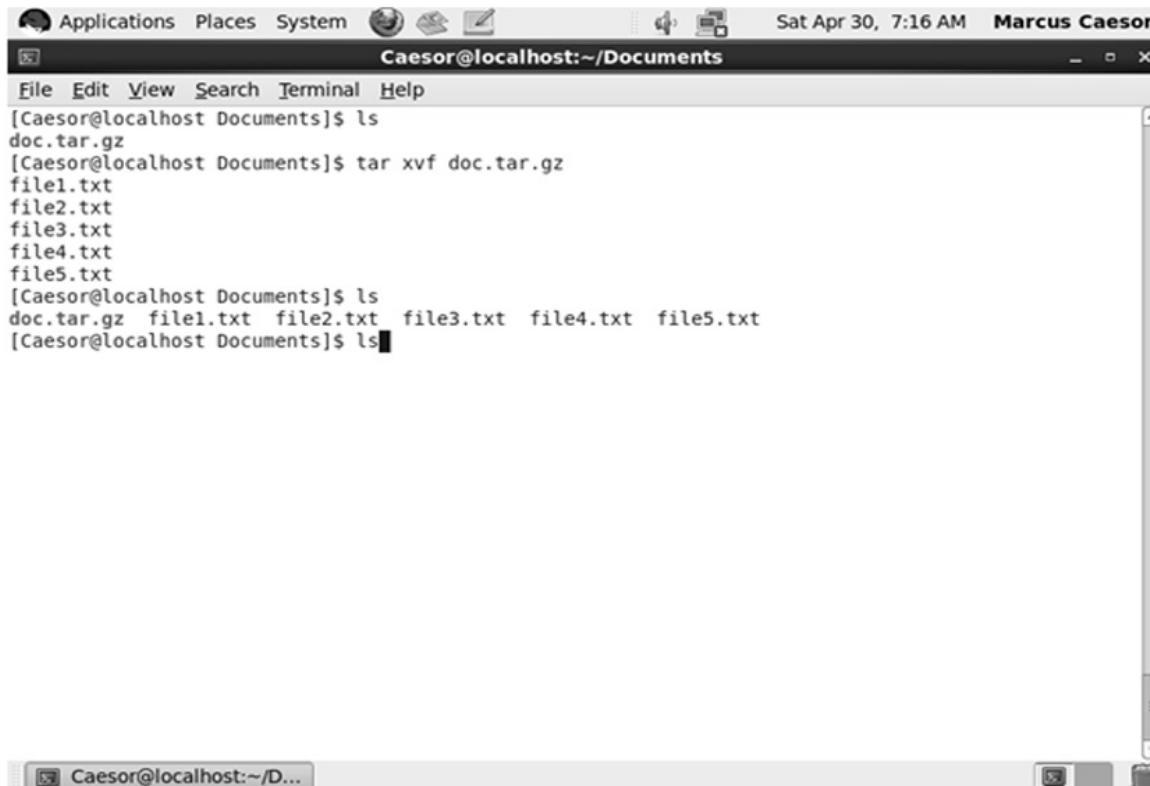
```
[Caesor@localhost Documents]$ ls  
doc.tar.gz  
[Caesor@localhost Documents]$ tar xvf doc.tar.gz  
file1.txt  
file2.txt  
file3.txt  
file4.txt  
file5.txt  
[Caesor@localhost Documents]$ ls  
doc.tar.gz file1.txt file2.txt file3.txt file4.txt file5.txt  
[Caesor@localhost Documents]$ ls
```

The terminal window has a standard Linux-style interface with a menu bar, a title bar, and a scrollable text area. The window title is "Caesor@localhost:~/Documents". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The title bar also displays the date and time "Sat Apr 30, 7:16 AM" and the user "Marcus Caesor". The scroll bar on the right side of the terminal window indicates that there is more text than can fit in the visible area.

Figure 7.8: Extracted Files

Step 8 - Execute the following command to view the files from the **.tar.gz** file without decompressing the file (Refer to Figure 7.9):

```
# tar tvf<filename>.tar.bz
```



The screenshot shows a terminal window titled "Caesor@localhost:~/Documents". The window has a standard Linux desktop interface with icons for Applications, Places, System, and a menu bar with File, Edit, View, Search, Terminal, and Help. The title bar also displays the date and time: "Sat Apr 30, 7:16 AM" and the user name "Marcus Caesor". The terminal content is as follows:

```
[Caesor@localhost Documents]$ ls
doc.tar.gz
[Caesor@localhost Documents]$ tar xvf doc.tar.gz
file1.txt
file2.txt
file3.txt
file4.txt
file5.txt
[Caesor@localhost Documents]$ ls
doc.tar.gz  file1.txt  file2.txt  file3.txt  file4.txt  file5.txt
[Caesor@localhost Documents]$ ls
```

Figure 7.9: Viewing Files from a .tar.gz File without Decompressing

Step 9 - Type **exit** and press ENTER to close the terminal.

Exercise - 4**Compressing and decompressing files in graphical mode**

Step 1 - Select the file or folder that has to be archived and compressed. Right-click the folder and click **Compress**. In the Compress dialog box (Refer to Figure 7.10), in the **Filename** **textbox**, type **personal**.



Figure 7.10: Compress Dialog Box

Step 2 - In the drop-down list, select the type of archive as shown in Figure 7.11. If the user has to use the combination of **tar** and **gzip**, choose the extension **.tar.gz**.



Figure 7.11: Selecting the Extension

Step 3 - Click Create.

Step 4 - An archive file named **personal.tar.gz** is created in the root directory of the **/personal** directory as shown in Figure 7.12.



Figure 7.12: Archived File

When a user has to restore the contents of the file, in the graphical mode, the user can right-click the **personal.tar.gz** file and select Extract Here.

Exercise - 5

Organizing directories and files

Note - Create a number of files with the following names in the home directory: file1.txt, file2.txt, file3.txt, file4.txt, file5.txt and doc1.txt.

In this exercise, users navigate through different directories, which are by default part of the operating system. To review the directory and file organization, a user must perform the following steps:

Step 1 - Log on to the RHEL system with the login credentials.

Step 2 - Click Applications → System Tools → Terminal to open the terminal.

Step 3 - At the command prompt, type the following command and press ENTER.

`ls`

This would display the current user's home directory.

Step 4 - Type the following command and press ENTER.

`cd ..`

This helps to navigate to the /home directory.

Step 5 - To display the /home directory structure, type `ls` command and press ENTER.

This would display the /home directory, which contains multiple user's home directories.

Note - RHEL /home directory contains multiple user profiles. When the terminal window is launched, it automatically navigates to the user's home directory.

Step 6 - Type the following commands and press ENTER.

`cd ..`

`ls`

Figure 7.13 shows the directory structure in RHEL.

```
[Caesor@localhost ~]$ ls
Desktop Document Documents Downloads Music Pictures Public Templates Videos
[Caesor@localhost ~]$ cd ..
[Caesor@localhost home]$ ls
[Caesor@localhost home]$ cd ..
[Caesor@localhost /]$ ls
bin cgroup etc lib media mnt newswap proc sbin srv tmp var
boot dev home lost+found misc net opt root selinux sys usr
[Caesor@localhost /]$
```

Figure 7.13: Directory Structure in RHEL

Note - There are a set of directories and subdirectories that are used for specific purposes. Some of these important directories are /bin, /usr/bin, /sbin, /usr/sbin, /tmp, /usr/local, /etc and /proc.

Exercise - 6

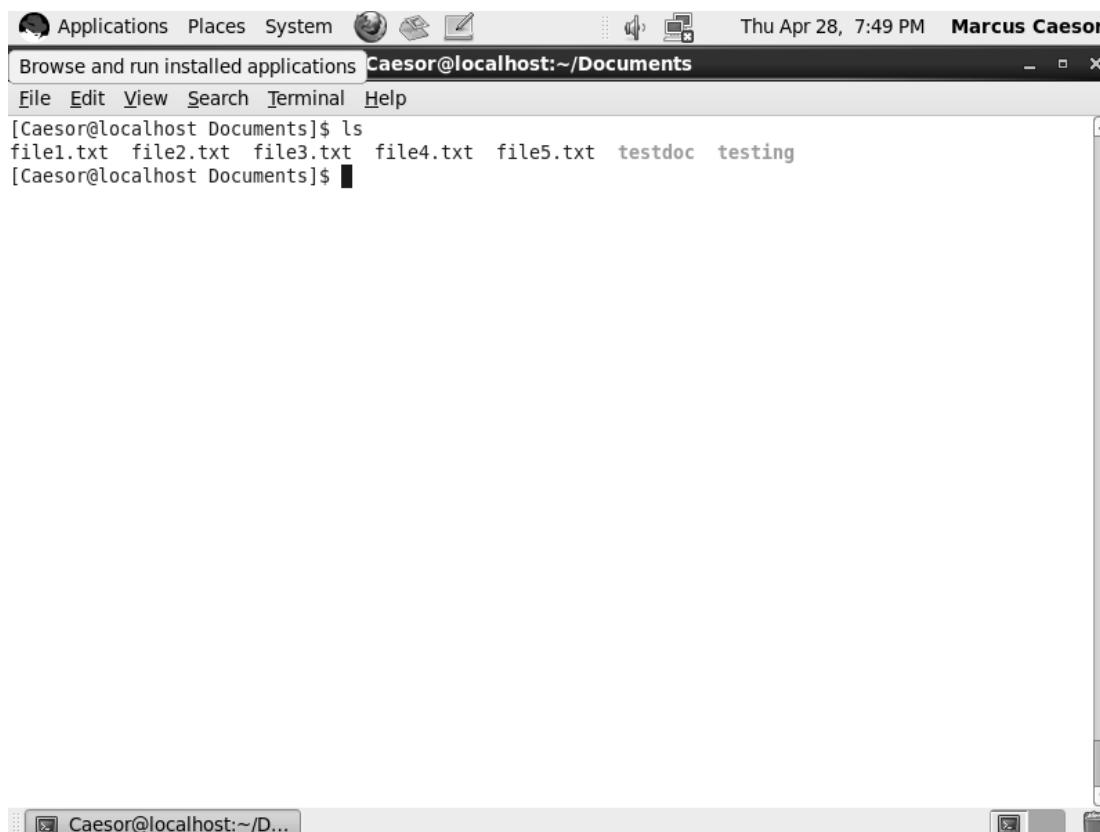
Listing directory contents

Note - Managing files can involve a number of operations. Managing files includes listing directory contents, copying and moving files and moving directories.

Step 1 - At the command prompt, type the following command and press ENTER.

`ls`

Figure 7.14 shows the listed files in a directory.



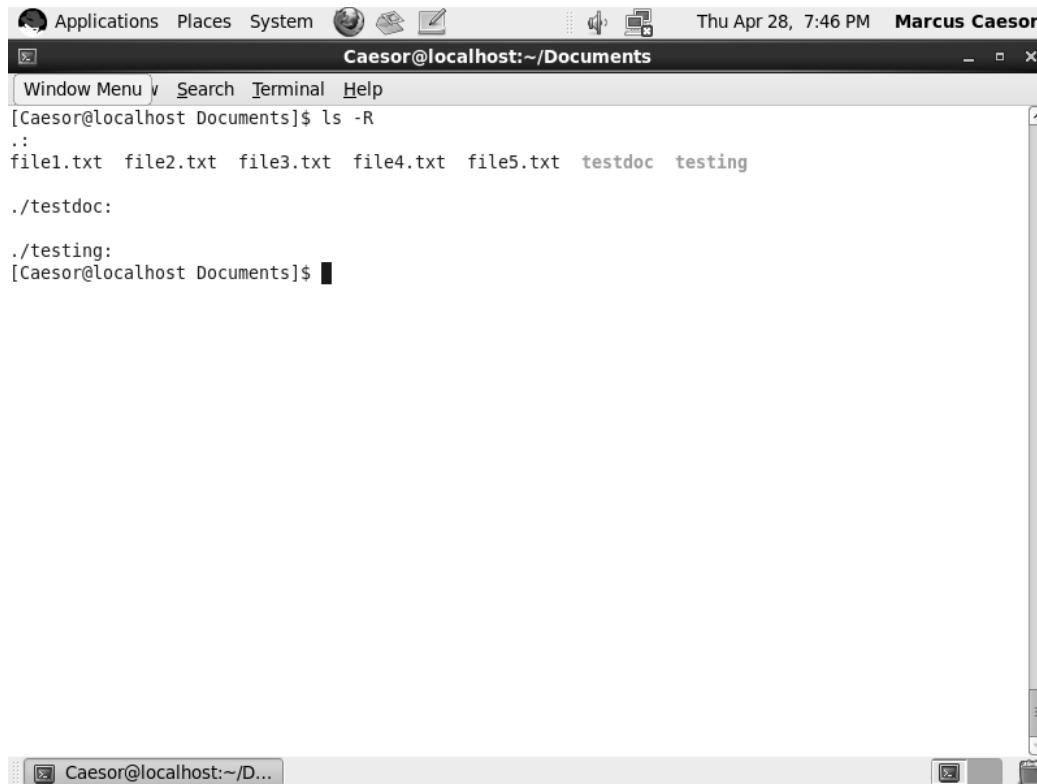
The screenshot shows a terminal window titled 'Caesor@localhost:~/Documents'. The window has a standard Linux desktop interface with icons for Applications, Places, System, and Dash. The title bar also displays the date and time as 'Thu Apr 28, 7:49 PM' and the user name 'Marcus Caesor'. The terminal window itself shows the command 'ls' being run, which lists several files: file1.txt, file2.txt, file3.txt, file4.txt, file5.txt, testdoc, and testing. The cursor is visible at the end of the command line.

Figure 7.14: Listed Files in Directory

Step 2 - To list the files within the subdirectories in user's home directory, type the following command and press ENTER.

`ls -R`

Figure 7.15 shows the output of the `ls -R` command.



The screenshot shows a terminal window titled "Caesor@localhost:~/Documents". The window contains the following text:

```
Applications Places System Window Menu Search Terminal Help
[Caesor@localhost Documents]$ ls -R
.:
file1.txt file2.txt file3.txt file4.txt file5.txt testdoc testing

./testdoc:

./testing:
[Caesor@localhost Documents]$
```

Figure 7.15: Output of `ls -R` Command

Exercise - 7

Copying and moving files

This exercise lists the steps for copying and moving files using the `cp` and `mv` command. To copy and move the files, the user has to perform the following steps:

Step 1 - At the command prompt, type `ls` command and press ENTER.

This should display the files, `file1.txt` to `file5.txt` and `doc1.txt`, created in the user's home directory.

Step 2 - Type the following command and press ENTER.

```
# cp file1.txt Documents
```

This copies the file, `file1.txt`, to the `/Documents` directory.

Step 3 - Type the following command and press ENTER.

```
# mv file2.txt Documents
```

This moves the file, `file2.txt`, to the `/Documents` directory.

Step 4 - Type `ls` command and press ENTER.

The listed files after moving the file are shown in Figure 7.16. The file2.txt file should no longer exist in this home directory.

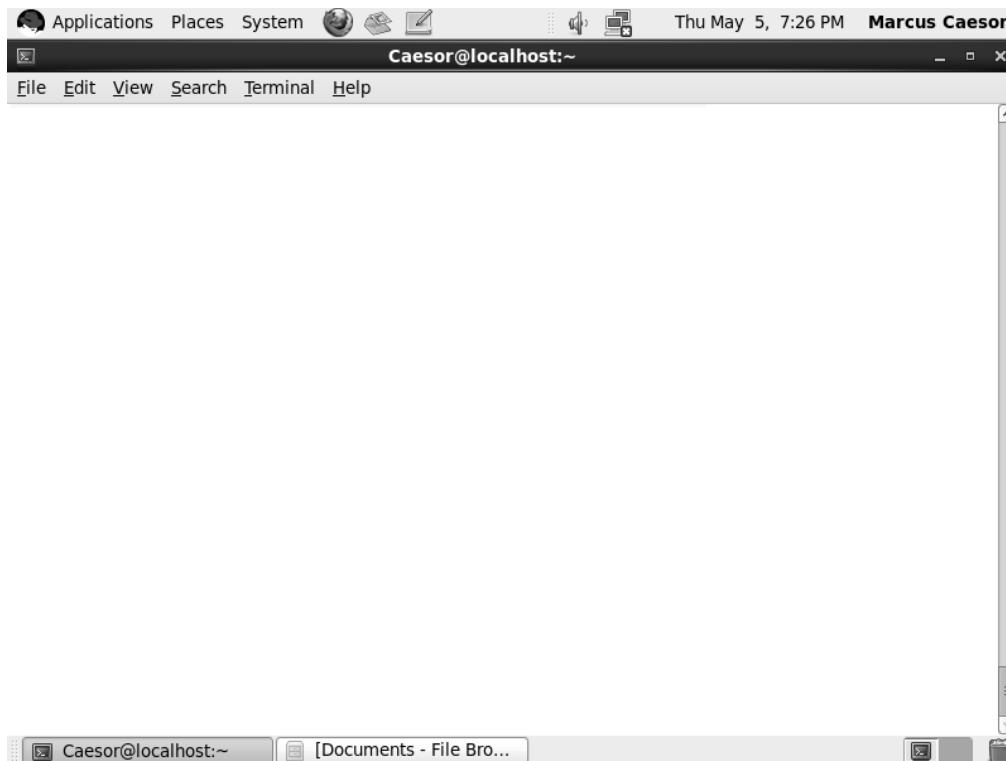


Figure 7.16: Listed Files

Moving Directories

This exercise demonstrates how to move a subdirectory to another subdirectory. To move a subdirectory, a user must perform the following steps:

Step 1 - At the command prompt, type the following command and press ENTER.

```
# mkdir mydata
```

This creates a directory with the name `mydata` within the user's home directory.

Step 2 - Type `ls` command to verify that the directory has been created.

Step 3 - Type the following command and press ENTER.

```
# mv mydata Downloads
```

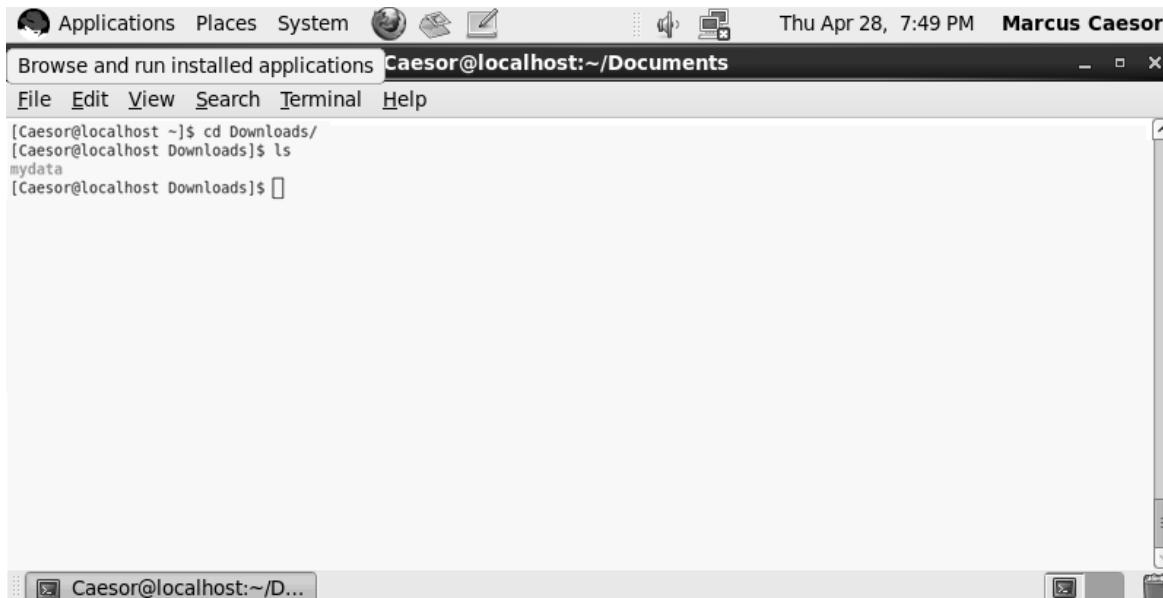
This moves the `/mydata` directory to the `/Downloads` directory from the user's home directory.

Step 4 - To verify that the directory no longer exists in the home directory, type `ls` at the command prompt.

Step 5 - Type `cd Downloads` and press ENTER.

Step 6 - Type `ls` and press ENTER.

Figure 7.17 shows the output of the `ls` command to move directories.



The screenshot shows a terminal window titled "Caesor@localhost:~/Documents". The window contains the following text:

```
[Caesor@localhost ~]$ cd Downloads/  
[Caesor@localhost Downloads]$ ls  
mydata  
[Caesor@localhost Downloads]$
```

Figure 7.17: Moving Directories

Step 7 - Type `clear` and press ENTER to clear the screen.

Exercise - 8

Creating a symbolic link

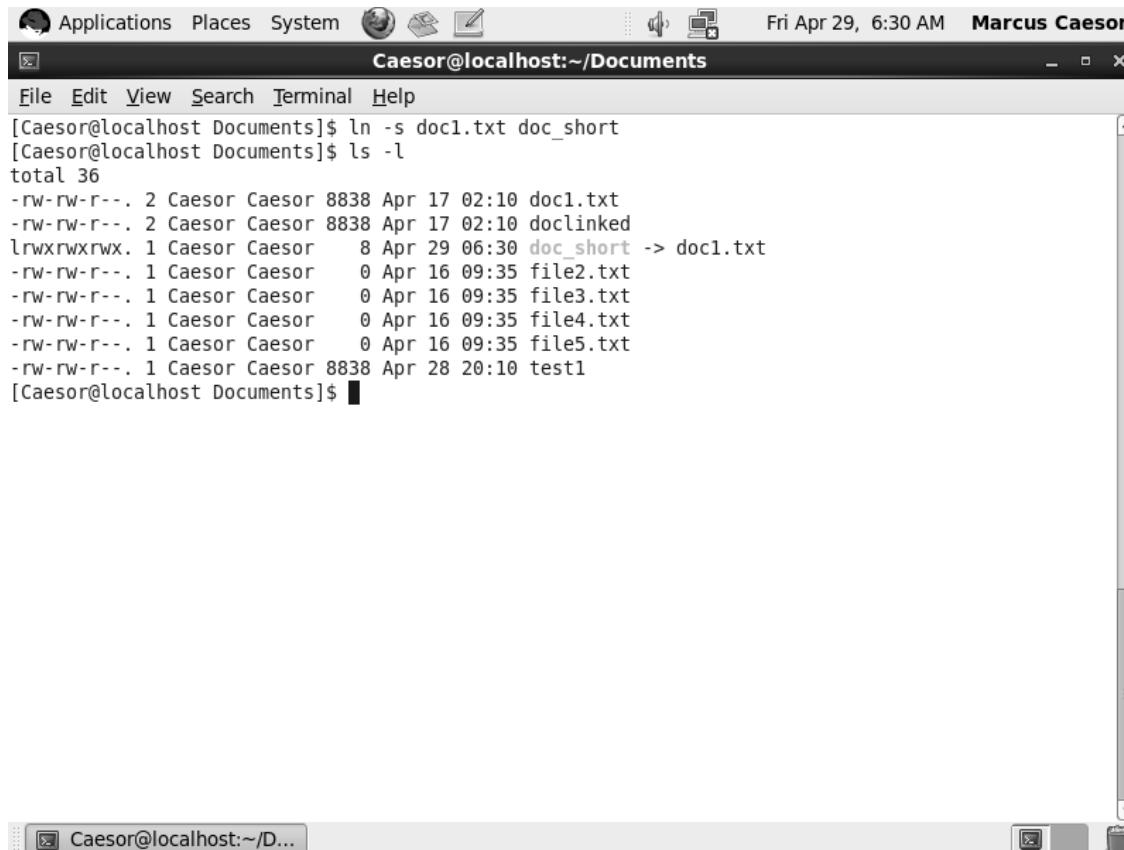
This exercise demonstrates how to create a symbolic link. To create a symbolic link, perform the following steps:

Step 1 - At the command prompt, type the following command and press ENTER.

```
ln -s doc1.txt doc_short
```

Step 2 - Type `ls -l` and press ENTER.

Figure 7.18 shows the creation of a symbolic link.



The screenshot shows a terminal window titled "Caesor@localhost:~/Documents". The window contains the following text:

```
[Caesor@localhost Documents]$ ln -s doc1.txt doc_short
[Caesor@localhost Documents]$ ls -l
total 36
-rw-rw-r--. 2 Caesor Caesor 8838 Apr 17 02:10 doc1.txt
-rw-rw-r--. 2 Caesor Caesor 8838 Apr 17 02:10 doclinked
lrwxrwxrwx. 1 Caesor Caesor     8 Apr 29 06:30 doc_short -> doc1.txt
-rw-rw-r--. 1 Caesor Caesor     0 Apr 16 09:35 file2.txt
-rw-rw-r--. 1 Caesor Caesor     0 Apr 16 09:35 file3.txt
-rw-rw-r--. 1 Caesor Caesor     0 Apr 16 09:35 file4.txt
-rw-rw-r--. 1 Caesor Caesor     0 Apr 16 09:35 file5.txt
-rw-rw-r--. 1 Caesor Caesor 8838 Apr 28 20:10 test1
[Caesor@localhost Documents]$
```

Figure 7.18: Creation of Symbolic Link

Exercise - 9

Creating a hard link

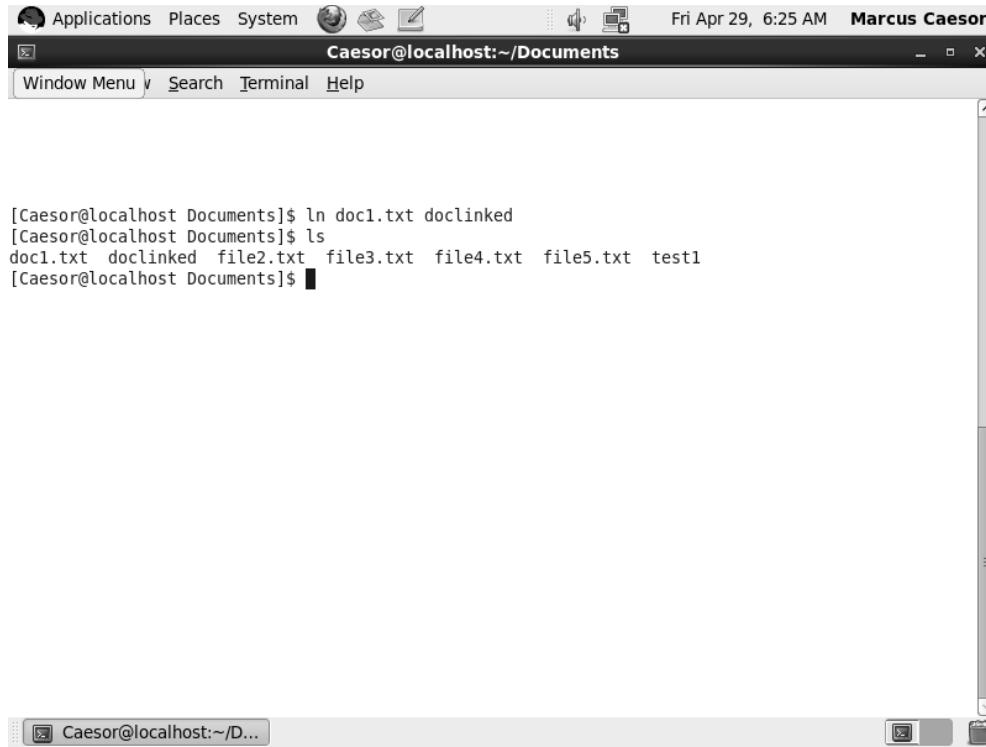
This exercise demonstrates how to create a hard link. To create a hard link, perform the following steps:

Step 1 - At the command prompt, type the following command and press ENTER.

```
# lndoc1.txt doclinked
```

Step 2 - Type ls and press ENTER.

Figure 7.19 shows the creation of a hard link.



The screenshot shows a terminal window titled "Caesor@localhost:~/Documents". The window contains the following command-line session:

```
[Caesor@localhost Documents]$ ln doc1.txt doclinked
[Caesor@localhost Documents]$ ls
doc1.txt doclinked file2.txt file3.txt file4.txt file5.txt test1
[Caesor@localhost Documents]$
```

Figure 7.19: Creation of Hard Link

Step 3 - Type exit and press **ENTER** to close the terminal.



Need
HELP
on a topic? = **FAQs**



www.onlinevarsity.com

8.1 Storage Considerations

RHEL supports a large number of file systems. Some of the key file systems are as follows:

- Second Extended file system (ext2)
- Third Extended file system (ext3)
- Fourth extended file system (ext4)
- XFS
- Swap
- New Technology File System (NTFS)
- Virtual File Allocation Table (VFAT)
- Global File System (GFS)
- Global File System 2 (GFS2)

The supports for a variety of file system are as follows:

- Backward compatibility
- Interoperability with other operating systems
- Expandability of RHEL operating system
- Large file systems
- Large supported file size

Each type of file system is capable of storing files and folders, typically known as directories. In RHEL, the root directory is the master directory that is the root of the file system. All other directories are known as the subdirectories.

Max Size Supported by File Systems

Some of the key file systems that it supports are listed in Table 8.1.

Feature	Ext2	Ext3	Ext4	XFS
Max Supported File Size	2 TB	2 TB	16 TB	16 TB
Max Supported Size	8 TB	16 TB	16 TB	100 TB
Maximum Number of Subdirectories	32,000	32,000	65,000	65,000
Maximum Depth of Symbolic Links	8	8	8	8

Table 8.1: Supported File Systems

In addition to the file systems listed in Table 8.1, RHEL supports some other file systems that are as follows:

- **Microsoft Disk Operating System (MS-DOS)** - This file system is typically used for floppy disks.
- **ISO 9660** - This file system is typically used for CDs.
- **GFS and GFS2** - These file systems are typically used for Storage Area Networks (SANs).

8.1.1 File System Structure

A file system structure is responsible for storing files and folders. An operating system, depending on the type of file system being used, utilizes these files and folders to provide relevant information to users and applications. Different file systems have different methods of organizing files and folders. Therefore, it largely depends on the file system to decide how a user or an application interacts with the files and folders. A common structure is mainly responsible for defining the permissions, names and locations for files and folders.

File system can divide files into different categories. These categories are as follows:

- **Shareable** - These are files that are on a local file system but remote hosts can access these files after they are shared. The local file system allows the files to be shared and accessed by remote hosts. A user typically requires permission to share these files on the local file system.
- **Unshareable** - These are files that cannot be shared and are locally stored and available on a file system.
- **Static** - These are files that cannot be changed. These are mostly known as system or application-related files that are used by the operating system or application. In some cases, an administrator can have the privilege to change these files, such as replace the file with a new version or modify the existing files.
- **Variable** - These files are documents that are typically created by users.

A common structure of a file system allows a controlled access to the files and folders. Controlled access is defined at the operating system level. Hence, the permissions defined on the files and folders are tightly integrated with the operating system. If a user is denied permission at the top-level folder, it is unlikely that the user can access files that exist under the top-level folders.

The methods that an operating system uses are as follows:

- Files are stored in the file system
- Permissions are defined by the operating system
- Permissions are assigned to a user or an application
- User accesses the files and folders based on the permissions defined

Overview of File System Hierarchy Standard (FHS)

The default file system structure for RHEL is FHS, which is a standard for all UNIX-like operating systems. This standard defines guidelines for the structure of files and directories placement. It mainly defines the standardization of files and directories across all UNIX-like operating systems.

The standardization guidelines define certain parameters for the files and directories that are as follows:

- Names
- Locations
- Permissions

The standardization guidelines help greater system interoperability, such as application designers and system integrators.

The FHS compliance has two key elements. These elements are as follows:

- The operating system should be compatible with other FHS-compliant operating systems.
- The operating system should have the capability to mount **/usr** read-only partition.

The FHS-compliant file systems have a few key directories. These are as follows:

- **/boot** - Contains the critical files that are required to boot the system. These are static files.
- **/dev** - Contains the device nodes that hold information on the connected devices and virtual devices. An example of device is **/dev/had**.

- **/etc/** - Contains the configuration files. These configuration files can be created by applications that are installed on the file system.

Other than the mentioned directories, a few more key directories are as follows:

- **/lib/**
- **/media/**
- **/mnt/**
- **/opt/**
- **/proc/**
- **/sbin/**
- **/srv/**
- **/sys/**
- **/usr/**
- **/var/**

8.1.2 Creating an ext3 File System

RHEL 6.0 uses ext4 as the default file system. However, in RHEL 5.0, ext3 was the default file system. Typically, users can define the file system during the installation of an operating system. However, when a new disk has been added or free space of an existing disk is to be used, users can create a partition and then format it with the ext3 file system.

Note - Users require superuser or root permissions to create a file system. Users can use either parted or **fdisk** utility to create a partition from the command-line. Users can also use the graphical tool from Applications → System Tools → Disk Utility.

Converting to an ext3 File System

There could be a possibility that users are still using a partition that is formatted with the ext2 file system. Users can upgrade the file system to ext3. To upgrade to ext3 file system, users must use the **tune2fs** command.

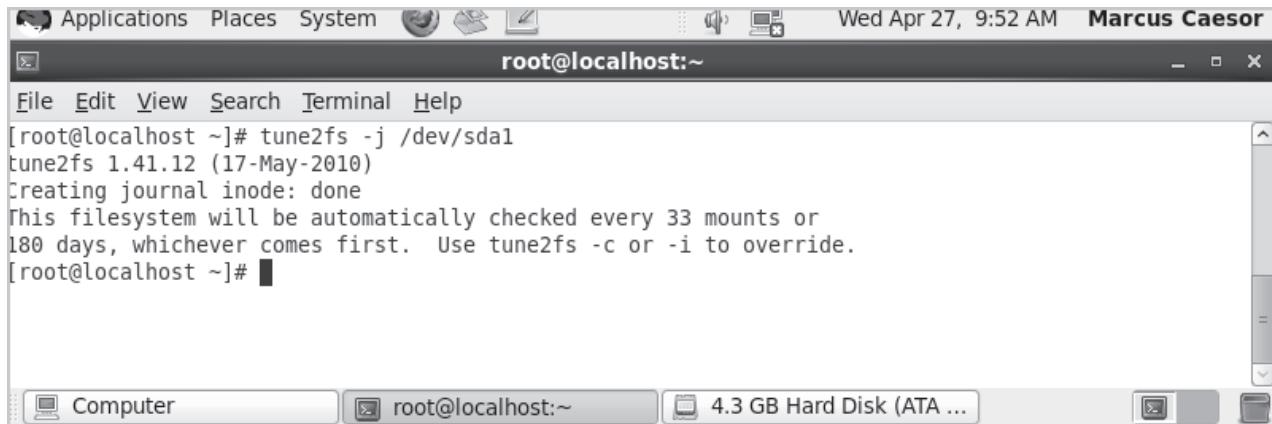
For example, a user can use the following command to convert an ext2 file system to ext3 file system:

```
tune2fs -j /dev/sdb1
```

The tune2fs command has a number of parameters. These parameters are as follows:

- -l
- -c max-mount-counts
- -e errors-behavior
- -f
- -i interval-between-checks
- -j
- -J journal-options
- -m reserved-blocks-percentage
- -o [^]mount-options[,...][-r reserved-blocks-count]
- -s sparse-super-flag
- -u user
- -g group
- -C mount-count
- -L volume-name
- -M last-mounted-directory
- -O [^] feature[,...]
- -T time-last-checked
- -U UUID

After a user executes the `tune2fs` command, the user gets the output as shown in Figure 8.1.



A screenshot of a terminal window titled "root@localhost:~". The window shows the command `tune2fs -j /dev/sda1` being run, followed by its output: "tune2fs 1.41.12 (17-May-2010)", "Creating journal inode: done", "This filesystem will be automatically checked every 33 mounts or 180 days, whichever comes first. Use tune2fs -c or -i to override.", and a final prompt "[root@localhost ~]#". The terminal is part of a desktop environment with a menu bar at the top and a dock at the bottom containing icons for Computer, root@localhost:~, and 4.3 GB Hard Disk (ATA ...).

Figure 8.1: Output of `tune2fs` Command

After the output is shown, the users can go back to the `/etc/fstab` file and change the file system from `ext2` to `ext3`.

→ Creating an ext4 File System

The ext4 file system is the default file system for RHEL 6.0. It provides many advantages over the previous file system, ext3. Two key advantages are increased file system size and support for larger file size up to 16 TB. The ext4 file system now uses extents instead of the block mapping scheme. The extents, introduced in ext4 file system, help to reduce metadata overhead when dealing with large files.

The ext4 file system also introduced improved system checks. This is useful if there are larger partitions.

Users can convert an existing ext3 file system to ext4 file system. To achieve this, execute the command that is as follows:

```
mkfs.ext4 /dev/device
```

Note - If the existing file system is already mounted, executing this command displays an error.

Figure 8.2 displays the output of `mkfs.ext4` command.

The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text:

```
[root@localhost ~]# mkfs.ext4 /dev/sda1
mke2fs 1.41.12 (17-May-2010)
/dev/sda1 is mounted; will not make a filesystem here!
[root@localhost ~]# umount /dev/sda1
[root@localhost ~]# mkfs.ext4 /dev/sda1
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
262144 inodes, 1048233 blocks
52411 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1073741824
32 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912, 819200, 884736

Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 28 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override
[root@localhost ~]#
```

The terminal window has a title bar with "root@localhost:~" and a status bar at the bottom showing "root@localhost:~" and "4.3 GB Hard Disk (ATA ...)".

Figure 8.2: Output of `mkfs.ext4` Command

8.2 Volumes

Volumes are defined on the physical hard disks that hold the RHEL operating system. Volumes are divided into partitions that can be of different types, such as boot partition or swap partition. Volumes can be created on a single physical hard disk or multiple hard disks and can be combined to create a single volume.

8.2.1 Basic and System Disk

A user can configures a disk either as a basic disk or system disk. When the user install RHEL on an x86 or x64 computer, there are three partitions that are created by the default partitioning layout.

These three partitions are as follows:

- The root partition (/)

→ **Swap partition**

→ **/boot partition**

The root partition (/) is on the basic disk and can be defined on the primary or logical partitions. This disk stores files and folders. The root partition can be defined as ext2, ext3 or ext4 partition. Swap partition works in coordination with the memory available to the operating system.

On the other hand, the /boot partition is the system disk for RHEL. When RHEL boots, the /boot partition is referred for necessary files. If this partition is corrupt or missing, the RHEL system cannot boot.

8.2.2 Logical Volumes

Logical volumes are based on physical volumes. They allow users to use a single or multiple physical storage devices using an abstraction layer. This layer hides the physical storage and allows the users and applications to view only the logical volumes. Assume that a user has two physical disks that are 100 GB each in size. Users can create a logical volume group that combines the space and applications or the users see combined space from all two disks. This provides flexibility to the administrator to manage a single logical volume group, rather than managing four physical disks.

The logical volume group can then be split into multiple smaller logical volumes for ease of use. These logical volumes can be used to create file system and mount points.

Figure 8.3 displays a logical volume group that consists of physical and logical volumes.

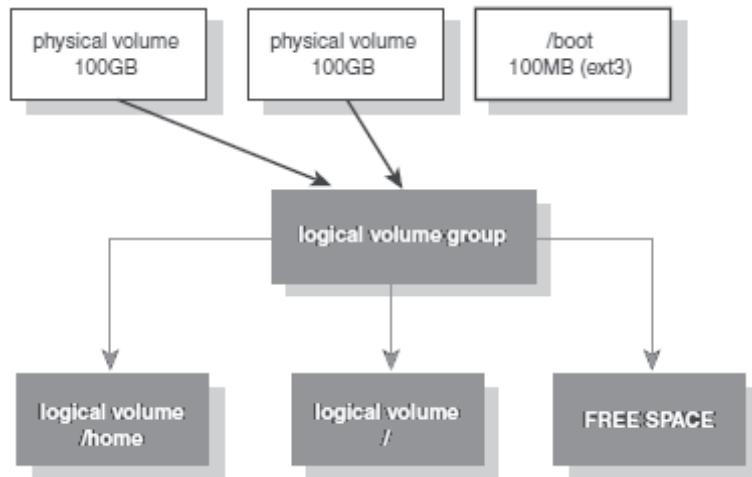


Figure 8.3: Logical Volume Group

The logical volumes have a number of benefits. These benefits are as follows:

- **Extension Across Disks** - Logical volumes can span across multiple physical disks, that otherwise can be used individually in most cases.

- **Resizing** - A user can resize the logical volumes without formatting the disks. This allows flexibility in terms of quickly resizing the logical volumes with simple commands.
- **Backups** - A user can use volume snapshots for backups. This helps in quick efficient backups without taking off the volume for backups.
- **RAID** - A user can create disk striping or mirrored volumes to increase throughput or redundancy.

A user can use Logical Volume Manager (LVM) to manage logical volume groups. LVM is used during the installation of RHEL to handle all partitions and mount points, except for **/boot** partition. The **/boot** partition is on a separate partition that cannot be a logical volume.

LVM has a number of tools to perform management of physical and logical volumes. These tools are as follows:

- pvcreate
- vgcreate
- vgextend
- vgreduce
- lvcreate
- lvextend
- lvremove
- vgdisplay
- lvdisplay
- pvscan

Using these tools, LVM can allow a user to perform a number of tasks that are as follows:

- Create physical and logical volumes
- Add or remove physical volume
- Create, remove or extend a logical volume

- Show properties of logical or physical volumes

8.2.3 Manage File System Attributes and Swap Space

The file system attributes define the type of partition, the layout and the storage process on the system. Using the file system attributes, a partition is defined in a disk drive. When a partition is created, its dimensions on the disk drive are defined. There are three key attributes that define a partition on a file system.

The three key attributes are as follows:

- **Partition Geometry** - The partition geometry defines the physical placement of a disk drive. Different factors combine to define the total disk size of a disk. Partition geometry is defined by four key factors. These factors are as follows:
 - Starting cylinders
 - Ending cylinders
 - Heads
 - Sectors
- **Partition Type** - A disk can primarily be formatted with three different kinds of partitions. These partitions are as follows:
 - **Primary** - There can be a maximum of four primary partitions on a disk.
 - **Extended** - This type of partition is created when a user requires more than four primary partitions. An extended partition can be further divided into more partitions.
 - **Logical** - An extended partition is usually divided into logical partitions. There can be multiple logical partitions within an extended partition.
- **Partition Type Field** - This parameter defines how the data is stored.

Swap Space

Swap space serves as an additional memory. The key role that swap space plays is to manage resources when the physical memory or RAM is full. In this case, the swap space serves as an alternate to RAM and handles all inactive pages that were blocking space in RAM. These inactive pages are then moved to the swap space and its space is freed up in RAM.

Depending on the memory requirement, swap space can be defined in different ways. A user can either configure the swap space to be on a dedicated partition or a file that is stored on the RHEL operating system. When a user uses automatic partition during the installation of RHEL, a swap partition is created. Before creating a swap space, first create a swap file and then convert the swap file to swap space. A swap file is a file that is created on the file system to act as additional memory for the operating system. It is used as an additional memory to store data when the operating system runs low on RAM.

Swap space is defined as double the space to the RAM available in the system. However, double amount of swap space is applicable if a system has less than 2 GB of RAM. If a system has more than 2 GB of RAM, define the same amount of swap space equal to the RAM available in the system.

In RHEL, the user must follow a number of steps. These steps are as follows:

1. Create swap space, which is usually a swap partition
2. Use mkswap to write special signature
3. Add entries to the **/etc/fstab** file
4. Finally, activate the swap partition using the swapon -a command

8.3 Check Your Progress

1. What are the default file systems in RHEL 6.0?

(A)	ext2	(C)	ext4
(B)	ext3	(D)	GFS

2. What are the maximum supported file sizes on ext4 file system?

(A)	16 GB	(C)	8 TB
(B)	2 TB	(D)	16 TB

3. Which are the default file system structures followed by RHEL?

(A)	GFS	(C)	DFS
(B)	EFS	(D)	FHS

4. Which types of partition are considered to be an alternate for RAM?

(A)	boot	(C)	swap
(B)	system	(D)	basic

5. Which commands are capable of converting an ext2 file system to ext3 file system?

(A)	tune2fs	(C)	mkfs.ext3
(B)	mkfs	(D)	fdisk

8.3.1 Answers

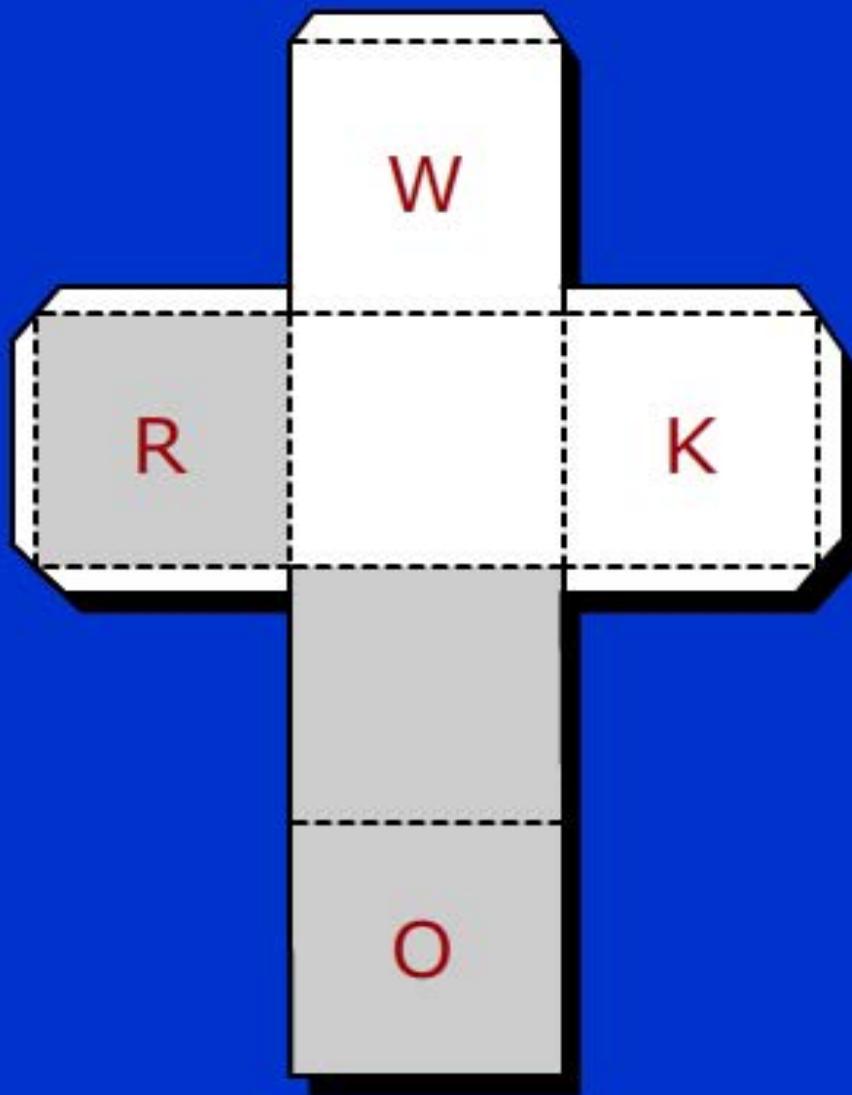
1.	C
2.	D
3.	D
4.	C
5.	A



Summary

- A file system structure is responsible for storing files and folders. Users require superuser or root permissions to create a file system. ext4 is the default file system based on File System Hierarchy Standard (FHS). Two key advantages of ext4 file systems are increased file system size and support for larger file size up to 16 TB. An existing ext3 file system can be converted to ext4 file system.
- Swap space is defined as double the space to the RAM available in the system. It is an alternate mechanism of storing cache that cannot be handled by RAM.

WORK



ASSIGNMENT:

Form the cube to read '**WORK**'.

"Practice does not make perfect. Only perfect practice makes perfect."

- Vince Lombardi

Practice the Practicals for Perfection @

www.onlinevarsity.com

Session - 9

Basic System Administration

Welcome to the Session, **Basic System Administration**.

This session explains system administration of RHEL operating system. The unit also explains archives creation and usage of the default Web browser in RHEL. In addition, the unit describes the steps to manage and monitor system resources and manage system software locally using Red Hat Network (RHN).

In this Session, you will learn to:

- Explain the features available for basic administration
- Explain network tasks such as printing and Web browsing
- Describe one time tasks and batch jobs
- Explain about Scheduling Tasks
- Describe Managing Installed Services
- Explain System Monitoring



9.1 Basic Administration

A system administrator has to manage multiple computers on a network. It is essential for a system administrator to know few key essential system-related tasks. This enables efficient management of RHEL computers.

9.1.1 Date and Time

When RHEL is installed in the Workstation mode, the date and time can also be configured post installation after the computer is restarted. A user, who is installing RHEL, can choose to either manually configure the time or synchronize it with the network-based time servers by selecting the Synchronize date and time over the network option on the Date and Time screen (Refer to Figure 9.1).

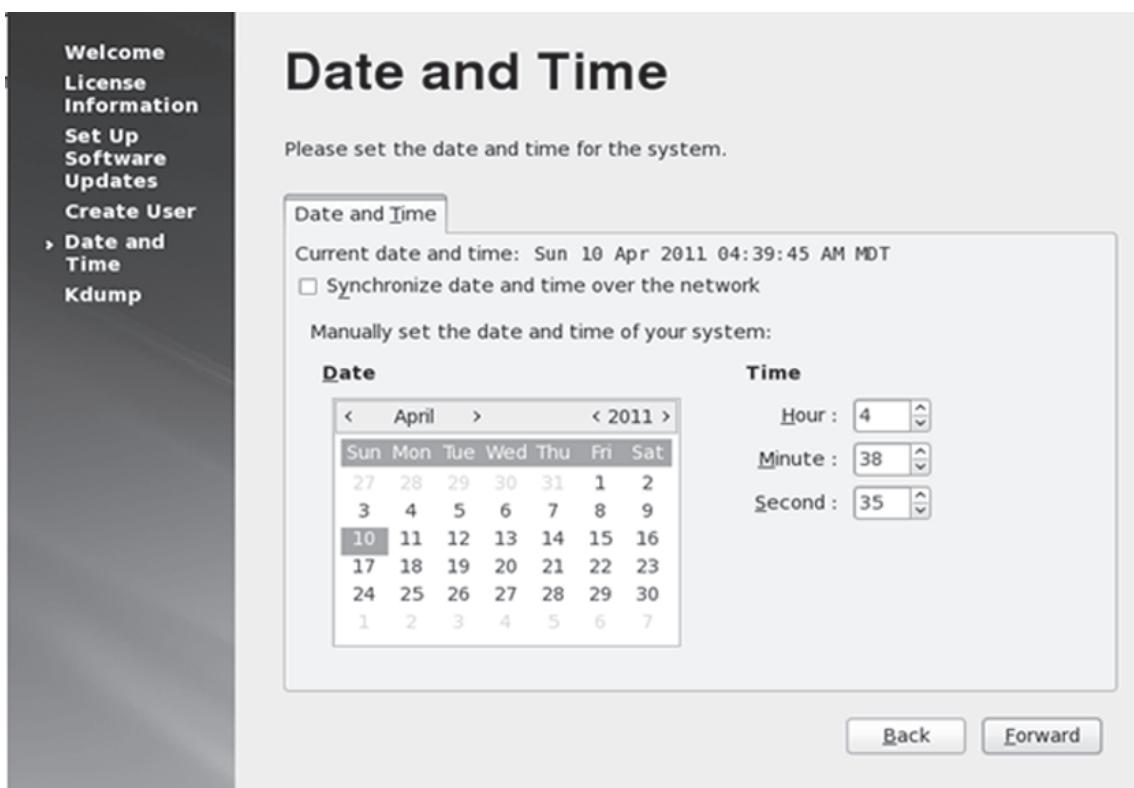


Figure 9.1: Date and Time Screen

9.1.2 System Logs

When an error is generated in an operating system, it is typically trapped in a log file. System administrators use the log file to find more details about the error and use these details to resolve the error.

RHEL also use logs to record certain amount of information. This information can be useful for system administrators for the following purposes:

- Resolve or review certain errors
- Adjust certain settings so that RHEL can perform at its optimum capacity

Some of the components that are recorded in the log files are as follows:

- Kernel
- Services
- Applications
- Security settings

RHEL uses a daemon named **rsyslogd** to trap certain information about the components. The **rsyslogd** daemon captures the information based on the directives that are defined in the **/etc/rsyslog.conf** file, which is a modular file that is organized into different sections containing different modules.

Some of the modules that are used in this file are as follows:

- **Input Modules** - Manages the gathering of information in form of messages
- **Output Modules** - Processes the messages into different formats
- **Filter Modules** - Filters the messages based on certain rules
- **Parser Modules** - Parses the messages
- **Message Modification Modules** - Changes the content of the **rsyslog** message
- **String Generator Modules** - Generates strings based on the received messages
- **Library Modules** - Manages loading of modules that are loadable

Figure 9.2 shows the contents of a sample **rsyslog.conf** file.

```
#rsyslog v3 config file

# if you experience problems, check
# http://www.rsyslog.com/troubleshoot for assistance

##### MODULES #####
$ModLoad imuxsock.so      # provides support for local system logging (e.g. via logger command)
$ModLoad imklog.so         # provides kernel logging support (previously done by rklogd)
#$ModLoad immark.so        # provides --MARK-- message capability

# Provides UDP syslog reception
#$ModLoad imudp.so
#$UDPServerRun 514

# Provides TCP syslog reception
#$ModLoad imtcp.so
#$InputTCPServerRun 514

##### GLOBAL DIRECTIVES #####
# Use default timestamp format
```

Figure 9.2: Contents of rsyslog.conf File

The captured logs are in plain text that can be either viewed using the **vi** or **Emacs** editor. Most of the log files are not accessible by a normal user. The root user privileges are required to open the log files. The log files that are marked with an **x (cross)** are protected log files that can be opened only by a root user.

Note - Some applications such as Apache, maintain their logs in their own directories and therefore, their log files cannot be available here.

Figure 9.3 shows the **/var/log** directory containing log files.

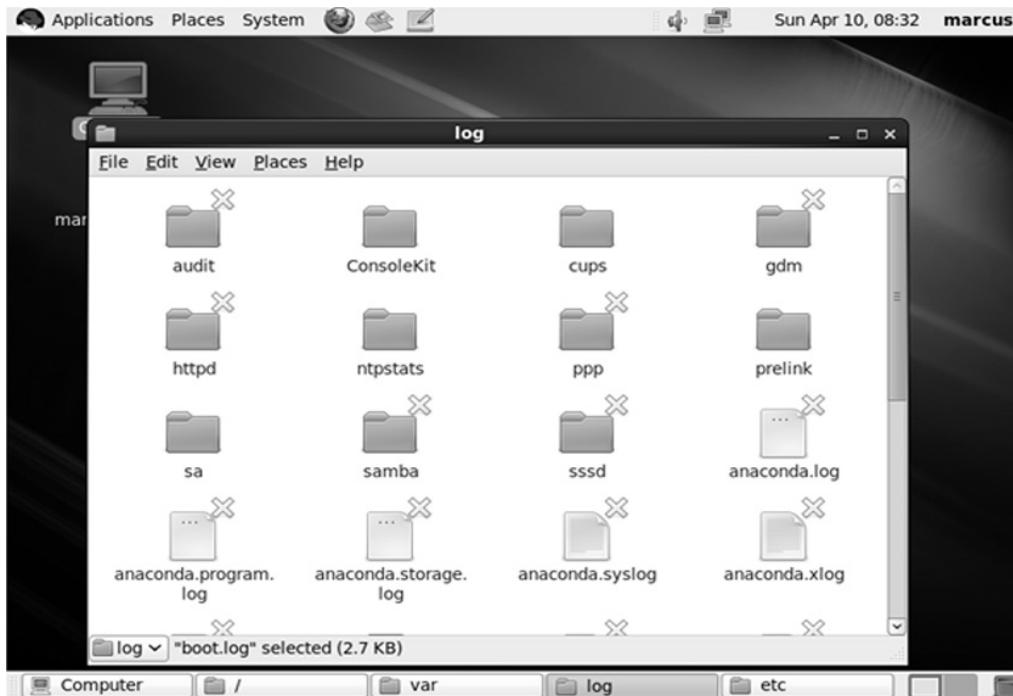


Figure 9.3: /var/log Directory Containing Log Files

Figure 9.4 shows a sample **boot.log** file.

```

%G%G>Welcome to %[0;31mRed Hat%[0;39m Enterprise Linux Server
Starting udev: %G%[60G[%[0;32m OK %[0;39m]

Setting hostname Marcus: %[60G[%[0;32m OK %[0;39m]

Setting up Logical Volume Management: 2 logical volume(s) in volume group "vg_marcus" now
active
%[60G[%[0;32m OK %[0;39m]

Checking filesystems
/dev/mapper/vg_marcus-lv_root: clean, 93838/1150560 files, 660815/4597760 blocks
/dev/sdal: clean, 38/128016 files, 45194/512000 blocks
%[60G[%[0;32m OK %[0;39m]

Remounting root filesystem in read-write mode: %[60G[%[0;32m OK %[0;39m]

Mounting local filesystems: %[60G[%[0;32m OK %[0;39m]

Enabling local filesystem quotas: %[60G[%[0;32m OK %[0;39m]

Enabling /etc/fstab swaps: %[60G[%[0;32m OK %[0;39m]

```

Figure 9.4: Sample boot.log File

A user can also use Yellowdog Updater, Modified (YUM) to install the gnome-system-log package that helps to view the log files and also filter each log based on the defined parameters.

Note - YUM is a free package management installation utility in RHEL.

9.1.3 Archiving Files

Users typically keep creating files and therefore files must be archived after a regular interval. Users can archive the files that are either old or not frequently changed. To archive files, a user can use the **tar** utility offered by RHEL.

The tar package must be installed on RHEL computer so that a user can use it. Typically, a user uses tar with either **gzip** or bzip2 utility, which offer compression. The **bzip2** is the newer compression utility and offers greater compression ratio as compared to the **gzip**.

The following are the benefits of using either of them in combination with tar:

- The data backup speeds up when it is compressed and at the destination location, the space is saved.
- When users retrieve the archive, they can always uncompress the same way as they compressed it.
- Users can combine multiple files within a single archive file.

When a user restores the files, the original directory structure is restored. The same directory structure is restored in the current directory from where the archive is being restored.

For example, a user has created an archive of **/personal** directory and the archive is named as **personal.tar.gz**. When the user copies the personal.tar.gz in the /temp directory, the data is restored in the **/temp/personal** directory.

The syntax of tar command is as follows:

```
#tar [OPTION...] [FILE]...
```

Note - To compress the file while archiving, the user must define the .gz extension along with the destination file name.

9.2 Scheduling Tasks

In Linux, jobs are the tasks that are performed on the RHEL system. Jobs can either run manually or automatically at a scheduled time. A job can be configured to run on a specific date, specific time or on a specific condition. There are a number of tasks that run by default in RHEL. These are usually the system tasks that must run to keep the system in running condition.

RHEL has a number of commands that can be used to schedule tasks. These are as follows:

- cron
- anacron
- at
- batch

9.2.1 cron

cron is a daemon that is used to schedule jobs. **cron** assumes that the systems are always in the running state. It is used to schedule recurring events. When a user logs on to the RHEL system, the **cron** daemon is automatically started from the **/etc/init.d** directory. **cron** is invoked each minute and checks the **crontabs** files for the scheduled jobs. The **crontabs** files are used by the **cron** daemon to schedule tasks. These files are stored in the **/var/spool/cron/crontabs/** directory.

The **cron** daemon must run continuously. If the RHEL system is turned OFF, the **cron** task fails if it is scheduled to run during the time when the system was switched OFF.

Following are the uses of **cron**:

- Keeping the log files in rotation
- Performing automatic clean up of system
- Rebuilding the **slocate** database
- Performing backups
- Cleaning up temporary directories
- Updating antivirus software

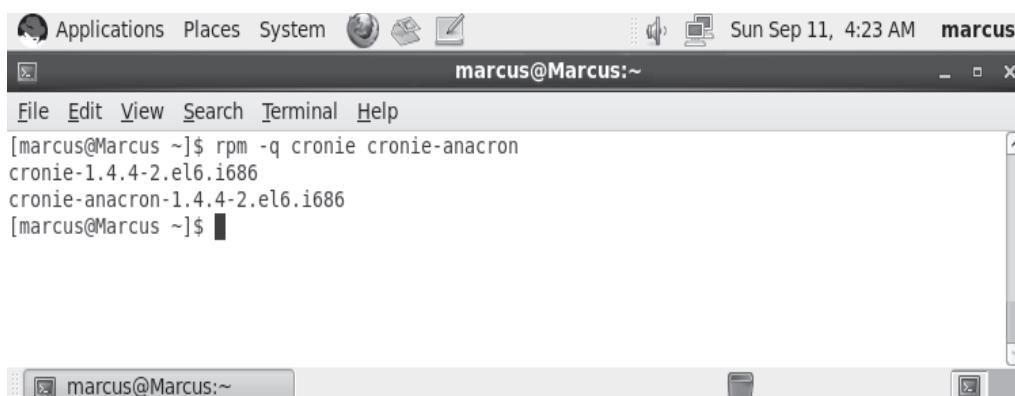
For a user to be able to use the **cron** command, the **cronie** Red Hat Package Manager (RPM) package must be installed on the RHEL system. This package is installed by default. A user can determine if the **cronie** package is installed on the RHEL system.

Following command is used to verify the **cronie** package:

```
rpm -q cronie cronie-anacron
```

The **cron** command is linked with the **crond** service. The **crond** service must be running to schedule a task in the RHEL system.

Figure 9.5 shows the result of the **cron** command.



The screenshot shows a terminal window titled "marcus@Marcus:~". The window contains the following text:

```
[marcus@Marcus ~]$ rpm -q cronie cronie-anacron
cronie-1.4.4-2.el6.i686
cronie-anacron-1.4.4-2.el6.i686
[marcus@Marcus ~]$
```

Figure 9.5: Result of the **cron** Command

Starting and Stopping the Service

A user can also verify the status of the **crond** service. The user can also start and stop the **crond** service and restart it if required.

Following command is used to verify the **crond** status:

```
/sbin/service crond status
```

To start the crond service, execute the following command:

```
/sbin/service crond start
```

To stop the service, execute the following the command:

```
/sbin/service crond stop
```

Note - By default, the **crond** service starts at the boot time. If it does not start, it is recommended that this service should be started at the boot time.

Exercise - 1

Starting, stopping and restarting the crond service

Note - User must have root permissions to edit the **rsyslog.conf** file. All other users have read permissions.

Step 1 - Log on to the RHEL workstation using the **username** and **password**.

Step 2 - Click **Applications** → **System Tools** → **Terminal** to open the command line terminal as shown in Figure 9.6.

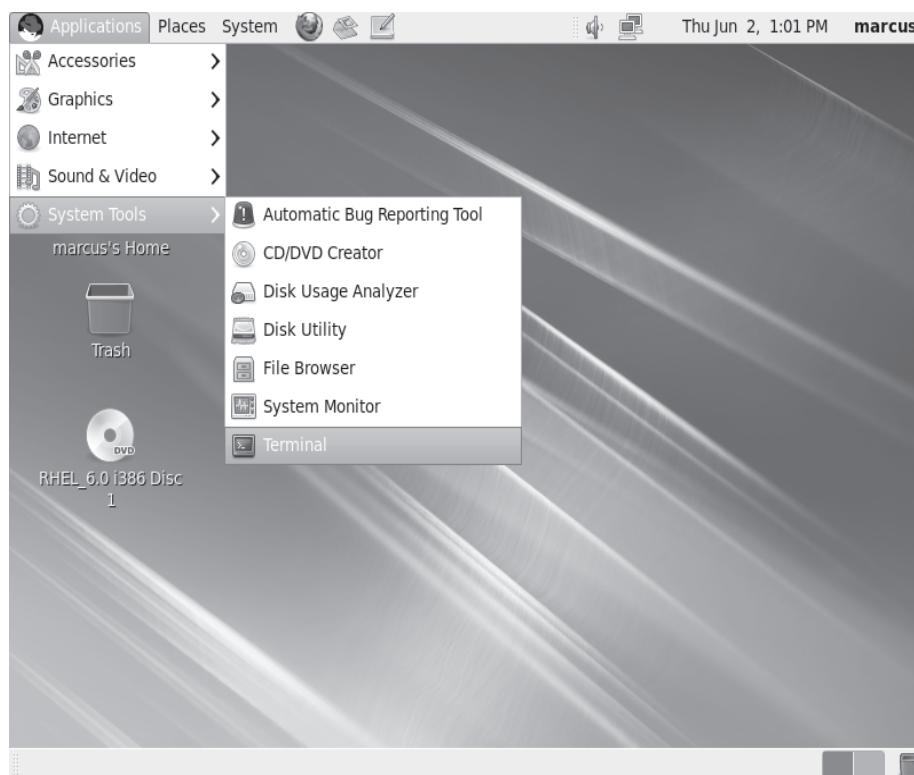


Figure 9.6: Terminal Option

The command line terminal appears as shown in Figure 9.7.

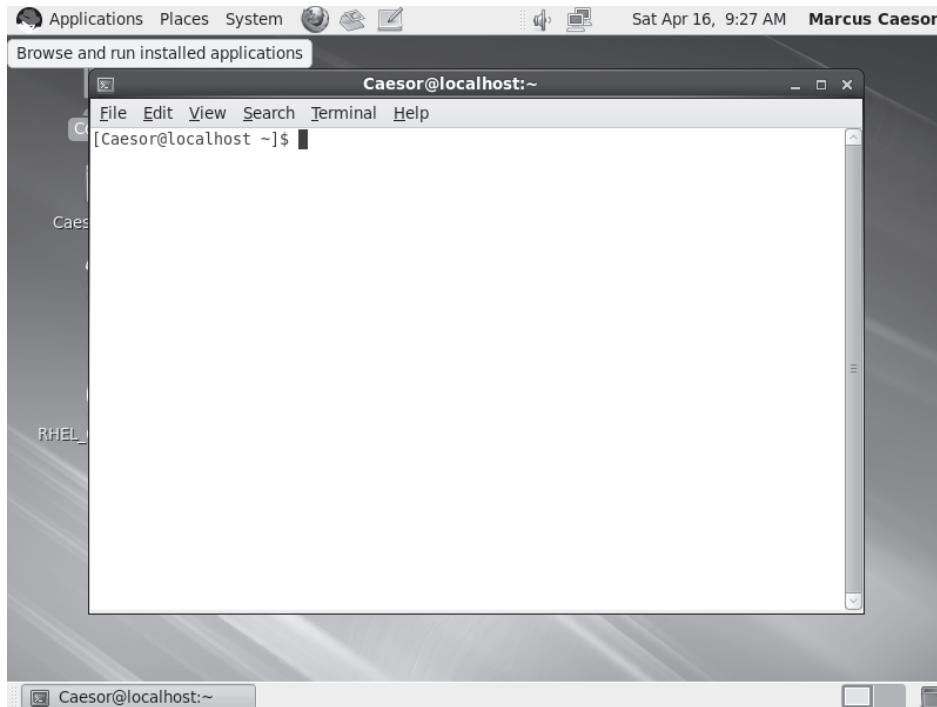


Figure 9.7: Command Line Terminal

Step 3 - Verify the installation of the **cronie** package by executing the following command (Refer to Figure 9.8):

```
rpm -q cronie cronie-anacron
```



Figure 9.8: Verifying the **cronie** Package Installation

Step 4 - Execute the following command to stop the **crond** service:

```
/sbin/service crond stop
```

Note - To stop a service, the user must have **superuser** privileges.

If not, the user command fails with the insufficient privileges as shown in Figure 9.9.



A screenshot of a terminal window titled "marcus@Marcus:~". The window shows the following command history:

```
[marcus@Marcus ~]$ rpm -q crontie cronie-anacron
crontie-1.4.4-2.el6.i686
crontie-anacron-1.4.4-2.el6.i686
[marcus@Marcus ~]$ /sbin/service crond stop
User has insufficient privilege.
[marcus@Marcus ~]$
```

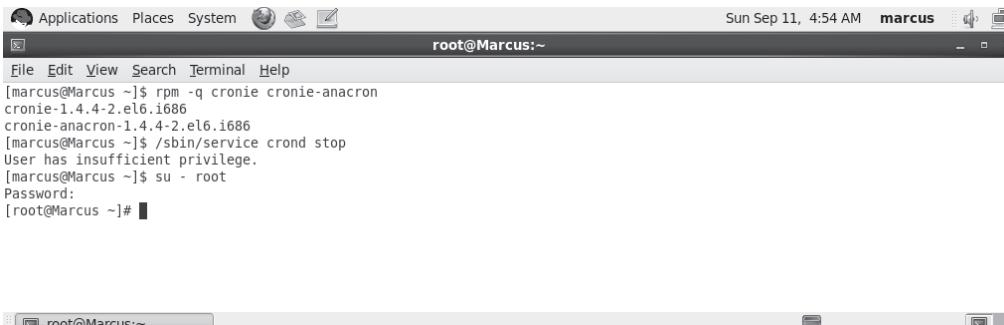
Figure 9.9: Insufficient Privileges

Step 5 - Execute the **su** command to gain administrative privileges of root user account.

Execute the following command, and then enter the password:

su - root

Figure 9.10 shows the result of this command.



A screenshot of a terminal window titled "root@Marcus:~". The window shows the following command history:

```
[marcus@Marcus ~]$ rpm -q crontie cronie-anacron
crontie-1.4.4-2.el6.i686
crontie-anacron-1.4.4-2.el6.i686
[marcus@Marcus ~]$ /sbin/service crond stop
User has insufficient privilege.
[marcus@Marcus ~]$ su - root
Password:
[root@Marcus ~]#
```

Figure 9.10: Result of the su-root Command

Step 6 - Execute the following command to stop the **crond** service (Refer to Figure 9.11):

/sbin/service crond stop



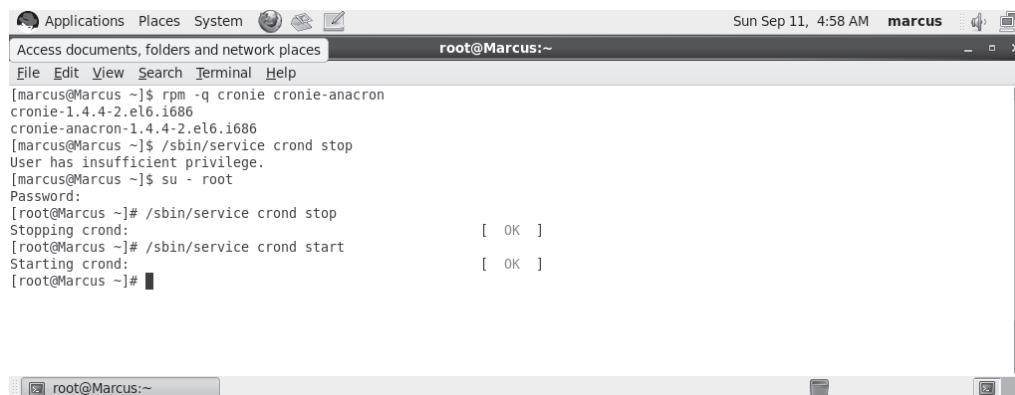
A screenshot of a terminal window titled "root@Marcus:~". The window shows the following command history:

```
[marcus@Marcus ~]$ rpm -q crontie cronie-anacron
crontie-1.4.4-2.el6.i686
crontie-anacron-1.4.4-2.el6.i686
[marcus@Marcus ~]$ /sbin/service crond stop
User has insufficient privilege.
[marcus@Marcus ~]$ su - root
Password:
[root@Marcus ~]# /sbin/service crond stop
Stopping crond: [ OK ]
[root@Marcus ~]#
```

Figure 9.11: Stopping the crond Service

Step 7 - Execute the following command to start the service (Refer to Figure 9.12):

```
/sbin/service crond start
```



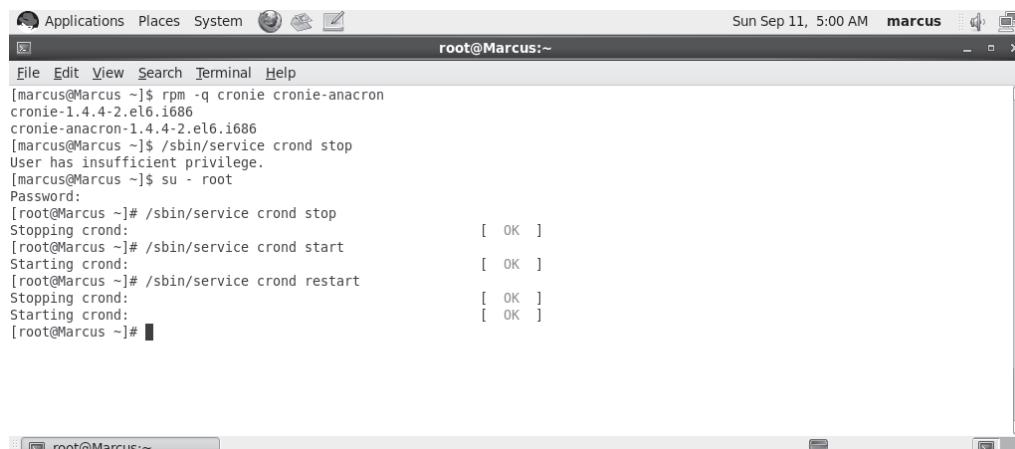
A screenshot of a terminal window titled "root@Marcus:~". The window shows the following command history:

```
[marcus@Marcus ~]$ rpm -q cronie cronie-anacron
cronie-1.4.4-2.el6.1686
cronie-anacron-1.4.4-2.el6.1686
[marcus@Marcus ~]$ /sbin/service crond stop
User has insufficient privilege.
[marcus@Marcus ~]$ su - root
Password:
[root@Marcus ~]# /sbin/service crond stop
Stopping crond: [ OK ]
[root@Marcus ~]# /sbin/service crond start
Starting crond: [ OK ]
[root@Marcus ~]#
```

Figure 9.12: Starting the **crond** Service

Step 8 - Execute the following command to restart the **crond** service (Refer to Figure 9.13):

```
/sbin/service crond restart
```



A screenshot of a terminal window titled "root@Marcus:~". The window shows the following command history:

```
[marcus@Marcus ~]$ rpm -q cronie cronie-anacron
cronie-1.4.4-2.el6.1686
cronie-anacron-1.4.4-2.el6.1686
[marcus@Marcus ~]$ /sbin/service crond stop
User has insufficient privilege.
[marcus@Marcus ~]$ su - root
Password:
[root@Marcus ~]# /sbin/service crond stop
Stopping crond: [ OK ]
[root@Marcus ~]# /sbin/service crond start
Starting crond: [ OK ]
[root@Marcus ~]# /sbin/service crond restart
Stopping crond: [ OK ]
Starting crond: [ OK ]
[root@Marcus ~]#
```

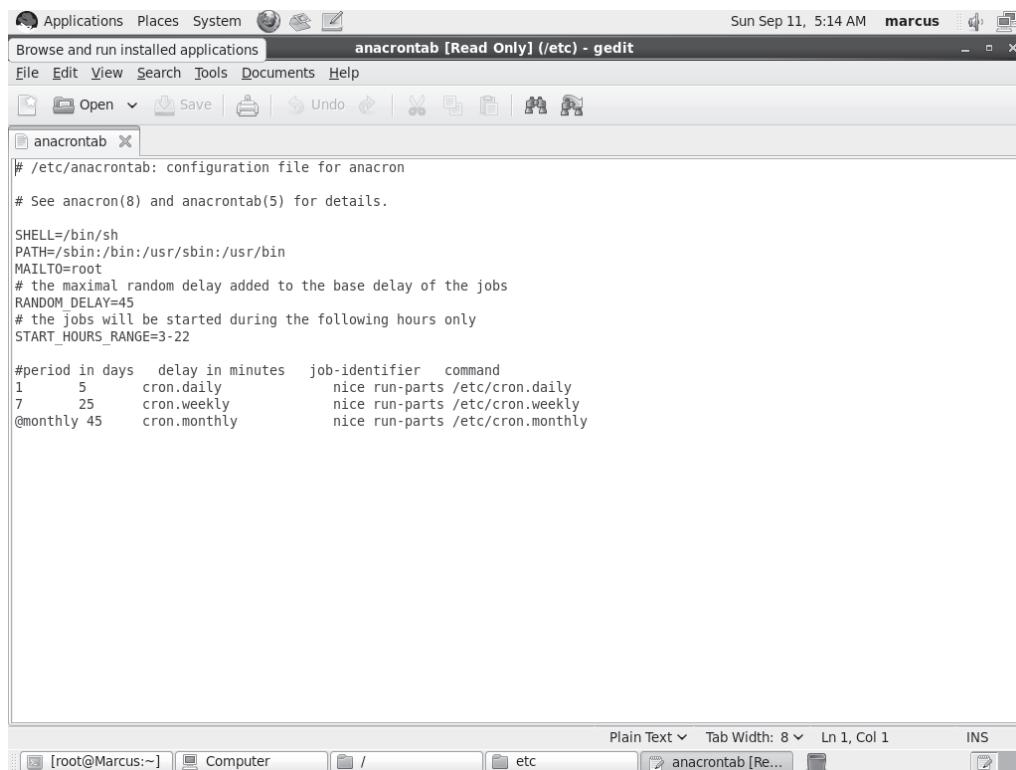
Figure 9.13: Restarting the **crond** Service

9.2.2 anacron

The **anacron** daemon is similar to **cron** with one distinct capability. To run **cron**, the system must be running continuously. On the other hand, **anacron** runs a job only once a day. If the system is turned OFF, the **anacron** daemon runs the job when the system is running the next time.

The configuration for the **anacron** daemon is in a file named **anacrontab**, which is located in the **/etc/** directory. It is important to note that only the root user can modify this file. Normal users do not have privileges to modify this file.

Figure 9.14 shows the sample of the **anacrontab** file.



The screenshot shows a Gedit text editor window titled "anacrontab [Read Only] (/etc) - gedit". The status bar at the top right indicates "Sun Sep 11, 5:14 AM marcus". The file content is as follows:

```
# /etc/anacrontab: configuration file for anacron
# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days    delay in minutes    job-identifier    command
1      5            cron.daily         nice run-parts /etc/cron.daily
7      25           cron.weekly       nice run-parts /etc/cron.weekly
@monthly 45        cron.monthly     nice run-parts /etc/cron.monthly
```

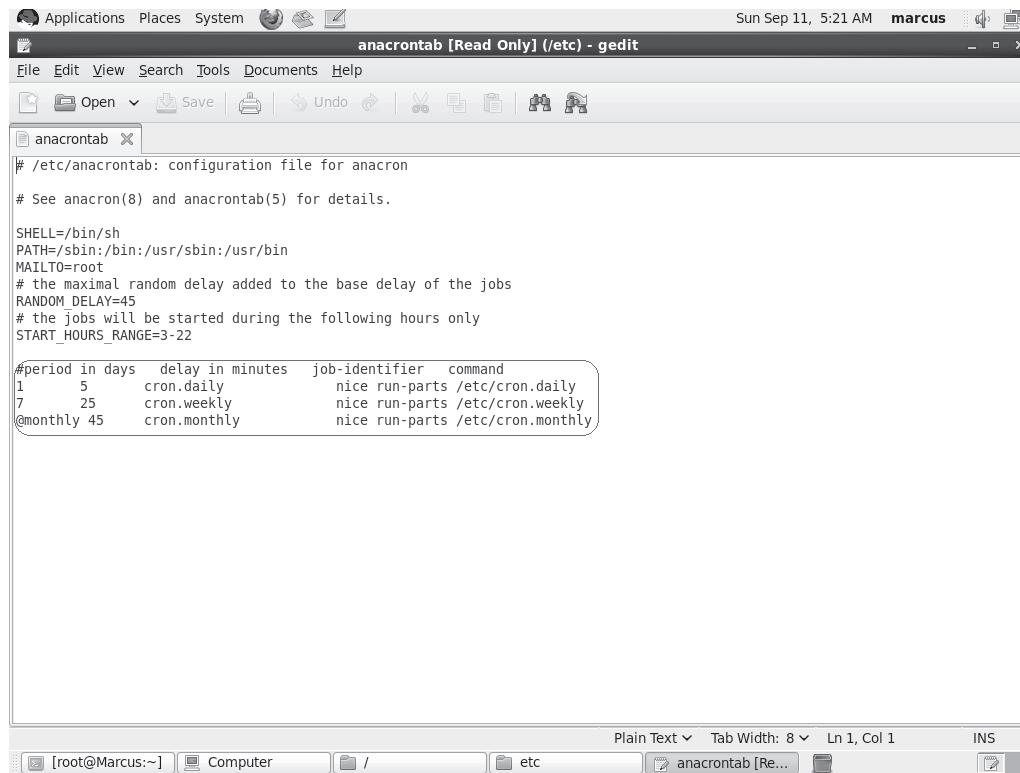
The status bar at the bottom right shows "Plain Text" and "Tab Width: 8". The bottom navigation bar includes tabs for "root@Marcus:~]", "Computer", "/", "etc", and "anacrontab [Re...]".

Figure 9.14: Sample of the anacrontab File

The scheduled tasks are specified in a specific format that is as follows:

period in days delay in minutes job-identifier command

Figure 9.15 shows three different scheduled tasks.



```

# /etc/anacrontab: configuration file for anacron
# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days    delay in minutes    job-identifier    command
1      5            cron.daily         nice run-parts /etc/cron.daily
7      25           cron.weekly        nice run-parts /etc/cron.weekly
@monthly 45          cron.monthly       nice run-parts /etc/cron.monthly

```

Figure 9.15: Scheduled Tasks

9.3 Managing One Time Tasks

There is always a possibility for the user to run a job only once. The job is required to run at a scheduled time. In this situation, the user can use the commands that are designed to run a task only once.

Following are the commands that can be used:

- ➔ **at**
- ➔ **batch**

9.3.1 at Command

The **at** command is used to run a specific job at a scheduled time. The **at** RPM package is required to run the **at** command. The **at** RPM package in RHEL is installed by default.

A user can verify if the package is installed with the following command (Refer to Figure 9.16):

```
rpm -q at
```

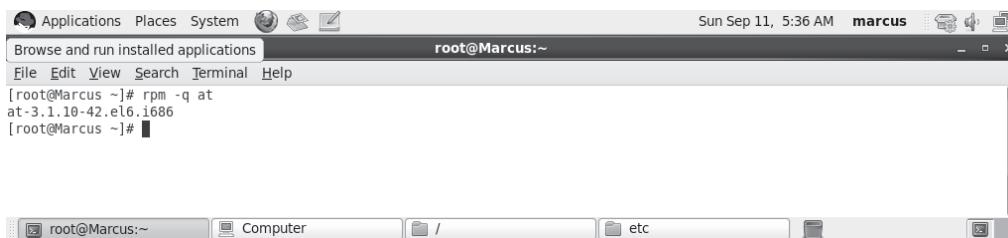


Figure 9.16: Verifying the `at` Package

A user can also verify if the `at` service is running by executing the following command (Refer to Figure 9.17):

`/sbin/service atd status`

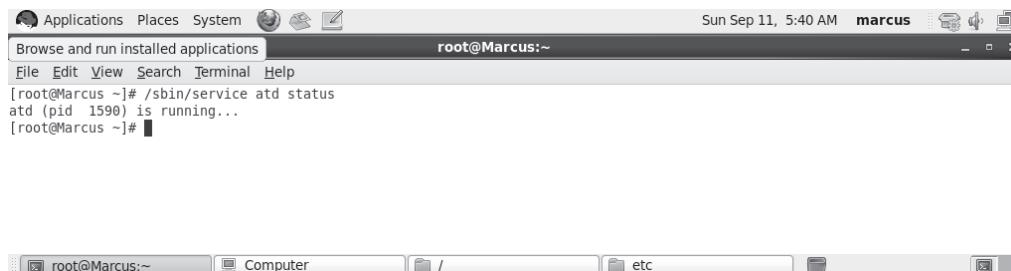


Figure 9.17: Status of the `atd` Service

The syntax for the `at` command is as follows:

`at time`

In the syntax, the time parameter signifies the time at which the tasks run.

When the `at` command is executed, the `at` prompt is displayed as shown in Figure 9.18.

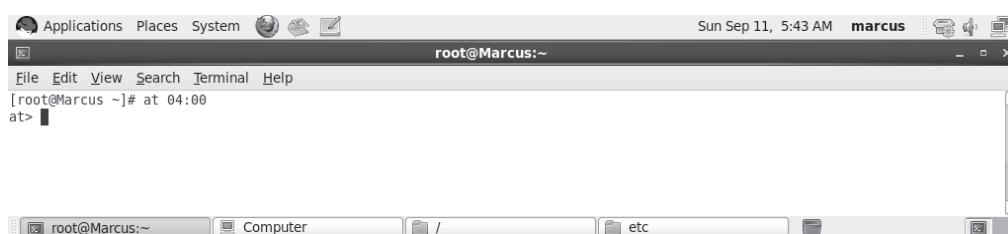


Figure 9.18: Displaying the `at` Prompt

A user can execute single or multiple commands on this prompt.

Note - Users can alternatively specify a batch file to execute multiple commands. After entering a single or multiple commands, the user must press **CTRL + D** to exit.

To view the pending commands, the user can execute the `atq` command. The output of this command shows the tasks that are scheduled to run but are pending at the time when the `atq` command runs.

9.3.2 batch Command

The **batch** command is similar to the **at** command. The **batch** command executes one-time tasks when the load is less than 0.8% on the server. The other characteristics are similar to the **at** command. It shows the **at** prompt when the user executes the **batch** command.

Figure 9.19 shows the **batch** command.



Figure 9.19: batch Command

Similar to the **at** command, the user can enter single or multiple commands, and then must press **CTRL + D** to exit.

9.4 The sysconfig Directory

The **sysconfig** directory is located in the **/etc** directory. The **sysconfig** directory stores the configuration files that are responsible for controlling the system configuration. There are two key components of this directory that are as follows:

→ Files

→ Directories

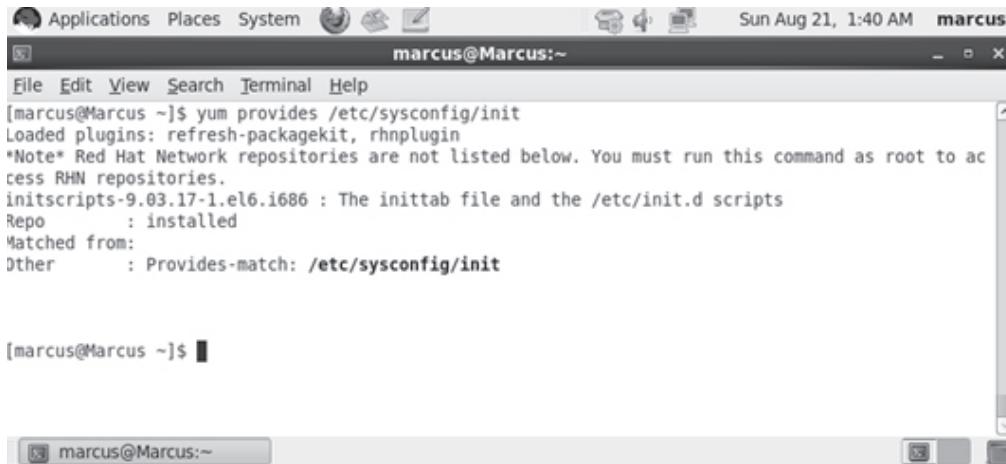
If a user requires complete information on the files and directories, the user can read through the **/usr/doc/initscripts-4.48/sysconfig.txt** file. The number of files and directories depend on the programs a user installs on a RHEL system. A user can locate a program name by referencing its configuration file. Following is the syntax to locate a program name to which a reference file is associated:

```
yum provides /etc/sysconfig/filename
```

If the user is already in the **/etc/sysconfig** directory, the following command should be executed:

```
yum provides filename
```

Figure 9.20 displays an example of the `yum` command.



The screenshot shows a terminal window titled "marcus@Marcus:~". The window contains the following text:

```
[marcus@Marcus ~]$ yum provides /etc/sysconfig/init
Loaded plugins: refresh-packagekit, rhnplugin
*Note* Red Hat Network repositories are not listed below. You must run this command as root to access RHN repositories.
initscripts-9.03.17-1.el6.i686 : The inittab file and the /etc/init.d scripts
Repo      : installed
Matched from:
Other     : Provides-match: /etc/sysconfig/init

[marcus@Marcus ~]$
```

Figure 9.20: Example of the `yum` Command

9.4.1 Files

The files in the `/etc/sysconfig` directory correspond to the programs installed in the RHEL system. The following files are normally found in the `/etc/sysconfig/` directory:

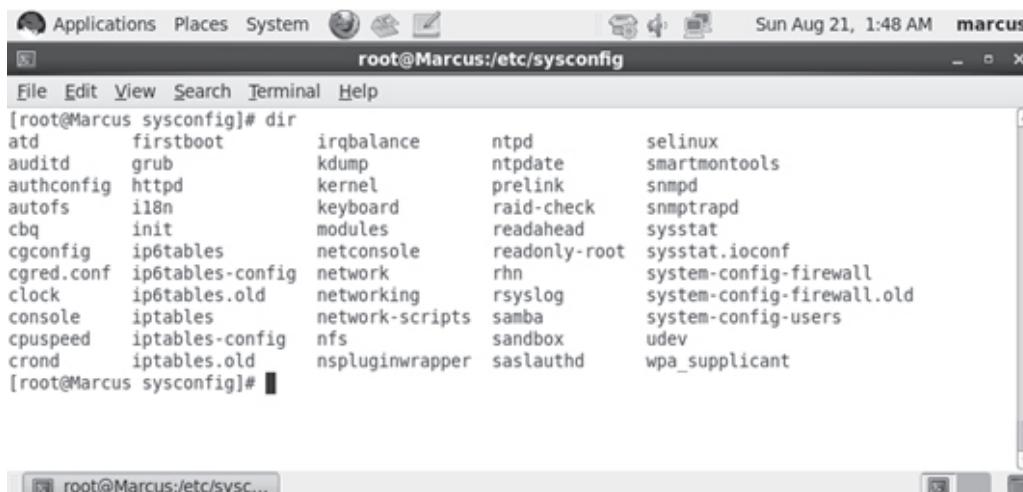
- ➔ amd
- ➔ apmd
- ➔ arpwatch
- ➔ authconfig
- ➔ autoofs
- ➔ clock
- ➔ desktop
- ➔ devlabel
- ➔ dhcpcd
- ➔ exim
- ➔ firstboot

- ➔ gpm
- ➔ harddisks
- ➔ hwconf
- ➔ i18n
- ➔ init
- ➔ ip6tables-config
- ➔ iptables-config
- ➔ irda
- ➔ keyboard
- ➔ kudzu
- ➔ mouse
- ➔ named
- ➔ netdump
- ➔ network
- ➔ ntpd
- ➔ pcmcia
- ➔ radvd
- ➔ rawdevices
- ➔ samba
- ➔ sendmail
- ➔ selinux

- spamassassin
- squid
- system-config-securitylevel
- system-config-users
- system-logviewer
- tux
- vncservers
- xinetd

If some of the files are missing from the **/etc/sysconfig** directory, it indicates that the program is not installed on the RHEL system.

Figure 9.21 displays the list of configuration files in the RHEL system.



```
[root@Marcus sysconfig]# dir
atd      firstboot      irqbalance    ntpd      selinux
auditd   grub          kdump        ntpdate   smartmontools
authconfig httpd        kernel       prelink   snmpd
autofs    i18n          keyboard     raid-check snmptrapd
cbq      init          modules      readonly-root sysstat
cgconfig ip6tables     netconsole   network   sysstat.ioconf
cgred.conf ip6tables-config   network   rhn      system-config-firewall
clock    ip6tables.old networking   rsyslog  system-config-firewall.old
console  iptables      network-scripts samba   system-config-users
cpuspeed iptables-config   nfs      sandbox  udev
crond    iptables.old nspluginwrapper saslauthd wpa_supplicant
[root@Marcus sysconfig]#
```

Figure 9.21: List of Configuration Files

Exercise - 1

Files in the /etc/sysconfig directory

Step 1 - Log on to the RHEL workstation using the **username** and **password**.

Step 2 - Click **Applications** → **System Tools** → **Terminal** to open the command line terminal as shown in Figure 9.22.

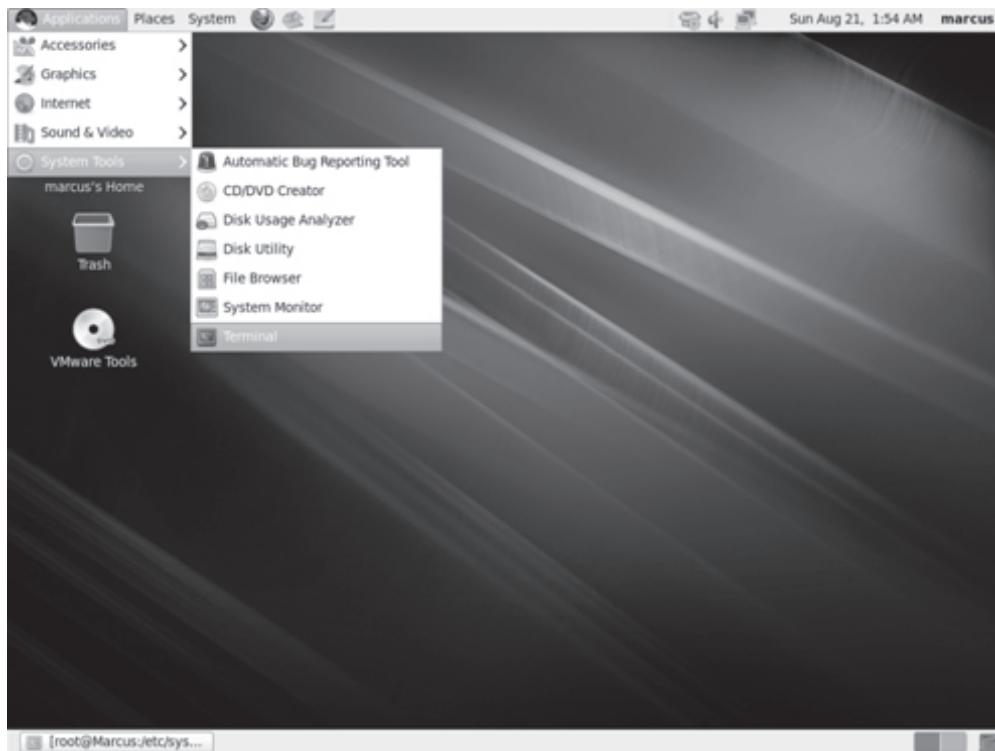


Figure 9.22: Navigating to Terminal

The command line terminal appears as shown in Figure 9.23.

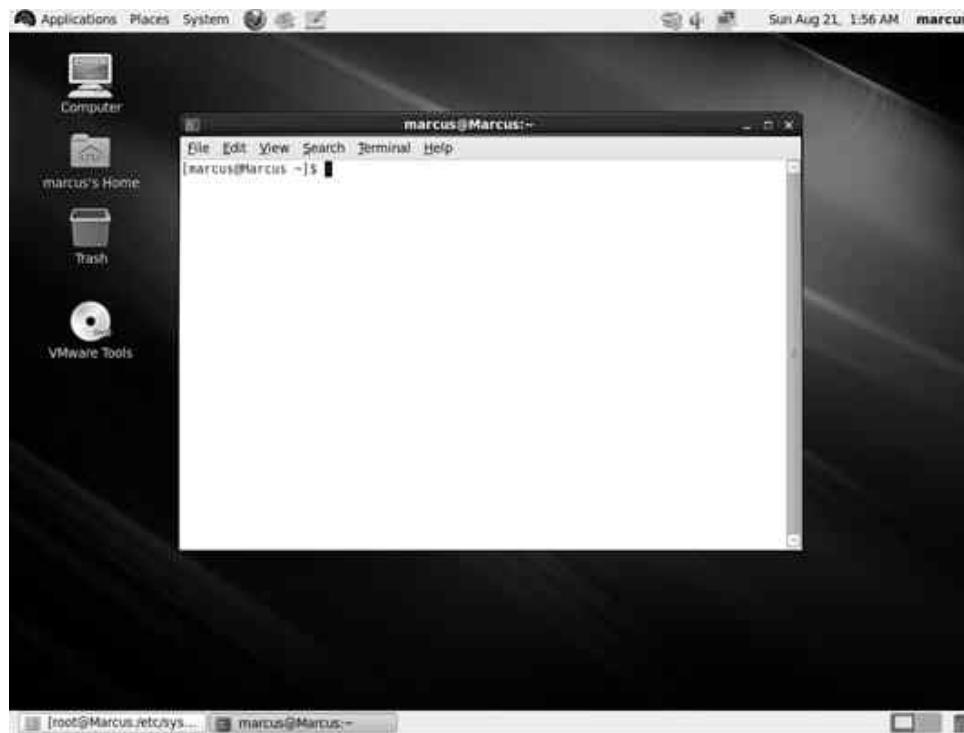


Figure 9.23: Command Line Terminal

Step 3 - Execute the following command:

```
cd /etc/sysconfig
```

The **/etc/sysconfig** directory is displayed as shown in Figure 9.24.



Figure 9.24: /etc/sysconfig Directory

Step 4 - View the package details for **authconfig** by executing the following command:

```
yum provides authconfig
```

Figure 9.25 shows the details of **authconfig**.



```
root@Marcus ~]# cd /etc/sysconfig
[root@Marcus sysconfig]# yum provides authconfig
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
authconfig-6.1.4-6.el6.i686 : Command line tool for setting up authentication from network services
Repo      : installed
Matched from:
Other     : Provides-match: authconfig

[root@Marcus sysconfig]#
```

Figure 9.25: authconfig Details

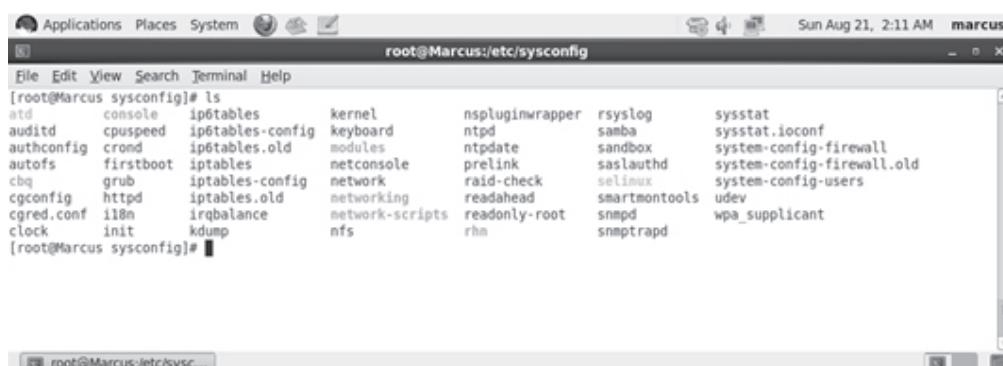
Package details for other files can also be displayed in a similar manner.

9.4.2 Directories

As compared to the files in the **/etc/sysconfig** directory, there are very few directories. Typically, the following directories are present:

- cbq
- networking
- network-scripts
- rhn

Figure 9.26 displays the common directories of the **/etc/sysconfig** directory.



```
root@Marcus ~]# cd /etc/sysconfig
[root@Marcus sysconfig]# ls
atd      console    ip6tables   kernel      nspluginwrapper  rsyslog      sysstat
auditd   cpuspeed   ip6tables-config  keyboard   ntpd          samba        sysstat.ioconf
authconfig  crond    ip6tables.old   modules    ntpdate       sandbox      system-config-firewall
autofs   firstboot  iptables      netconsole  prelink      saslauthd    system-config-firewall.old
cbq      grub       iptables-config  network    raid-check   selinux      system-config-users
cgroup   httpd      iptables.old   networking  readonly-root smartmontools udev
cgred.conf  i18n     irqbalance   nfs           rhn          snmpd       wpa_supplicant
clock    init       kdump        nfs           rhn          snmptrapd
[root@Marcus sysconfig]#
```

Figure 9.26: Common Directories

9.5 System Processes

A system process is an executing instance, which can be a command, an application or any other running task on the RHEL system. A new process is created when a command is executed or a program is started on the RHEL system. The Linux kernel is responsible for creating and managing the processes. When the Linux kernel creates a process, the kernel allocates system resources to the process. System resources exist in the system, such as CPU and memory. The system resources are prioritized by the kernel based on the state of the process. If the process is in the active state, the kernel allocates more system resources than the process that is in the idle state.

Allocation of resources is also dependent on the time frame. A process is assigned system resources for a specific time period. When the time period collapses, the system resources are assigned to the next waiting process.

The kernel performs different tasks for managing processes. Following are the tasks that the kernel performs on processes:

- Creation of Processes
- Execution of Processes
- Prioritization of Processes
- Suspension of Processes
- Termination of Processes

A process can be an independent or can have a sub process that runs along with the process. In this case, the process is called the parent process and the sub process is called the child process. The sub processes are typically created by the parent process. The relationship between the parent and the child process can be one to one or one to many. One parent process can either have single child process or multiple child processes.

9.5.1 Gathering System Information

To be able to configure and use an RHEL system to its optimal level, a user should know about the system processes in detail. Each system that runs RHEL has specific set of system resources that run these processes. These system resources are a combination of CPU, memory and hard disk space. It is essential for the user to know about the system and its resources. This knowledge helps the user to utilize the system resources to its optimal level.

After a user gathers the system information, only then the user is able to configure the system to be used to its full capacity.

For example, if the user is unaware of the total hard disk space, the user is unable to find how much space is being utilized or how much space is free. Therefore, some basic information should be known to the user.

Following are some pointers that a RHEL user should know when using the system:

- CPU clock speed
- Total memory installed and free memory
- Total hard disk space and free space available
- Partitions in the hard drive
- Running processes

9.6 Viewing System Processes

There are a number of commands that a user can use to view system processes. These commands are as follows:

- `ps ax`
- `top`
- `ps aux`

9.6.1 `ps ax` Command

The `ps ax` command is used to display the current set of system processes. This output list that generates the process list also includes processes that are owned by other users. The `ps ax` command generates a long list of processes. It is wise to pipe the command with the `less` pipe.

Figure 9.27 shows output of the `ps ax | less` command.

```

Applications Places System Mon Aug 15, 10:29 AM marcus
File Edit View Search Terminal Help
root@Marcus:~
PID TTY      STAT   TIME  COMMAND
1 ?        Ss     0:01 /sbin/init
2 ?        S      0:00 [kthreadd]
3 ?        S      0:00 [migration/0]
4 ?        S      0:00 [ksoftirqd/0]
5 ?        S      0:00 [watchdog/0]
6 ?        S      0:00 [events/0]
7 ?        S      0:00 [cpuset]
8 ?        S      0:00 [khelper]
9 ?        S      0:00 [netns]
10 ?       S      0:00 [async/mgr]
11 ?       S      0:00 [pm]
12 ?       S      0:00 [sync_supers]
13 ?       S      0:00 [bdi-default]
14 ?       S      0:00 [kintegrityd/0]
15 ?       S      0:00 [kblockd/0]
16 ?       S      0:00 [kacpid]
17 ?       S      0:00 [kacpi_notify]
18 ?       S      0:00 [kacpi_hotplug]
19 ?       S      0:02 [ata/0]
20 ?       S      0:00 [ata_aux]
21 ?       S      0:00 [ksuspend_usbd]
22 ?       S      0:00 [khubd]
23 ?       S      0:00 [kseriod]
25 ?       S      0:00 [khungtaskd]
26 ?       S      0:00 [kswapd0]
27 ?       SN     0:00 [ksmd]
28 ?       S      0:00 [aio/0]
:

```

Figure 9.27: Output of the `ps ax | less` Command

9.6.2 top Command

The `top` command provides information about the current processes. It also provides the information in real-time, which implies that the process list is dynamic and keeps updating when the command is in the execution mode. The command also provides information about the memory and CPU usage in the output. The `top` command has a number of parameters. Table 9.1 displays the parameters of the `top` command.

Parameter	Description
Space	Refreshes screen.
h	Displays help screen.
k	Kills a process.
n	Changes the number of processes that are displayed on the screen. User is prompted for the number.
u	Sorts the output by username.
M	Sorts the output by memory usage.
P	Sorts the output by the CPU usage.

Table 9.1: Parameters of the `top` Command

Figure 9.28 displays the output of the `top` command.

```

top - 10:36:57 up 2:59, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 140 total, 1 running, 138 sleeping, 1 stopped, 0 zombie
Cpu(s): 0.7%us, 0.3%sy, 0.0%ni, 99.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1031320k total, 606508k used, 424812k free, 94668k buffers
Swap: 2064376k total, 0k used, 2064376k free, 313396k cached

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1712 root 20 0 63764 18m 7548 S 0.3 1.8 0:11.02 Xorg
1727 root 20 0 21900 2820 2024 S 0.3 0.3 0:00.19 console-kit-dae
5817 root 20 0 2660 1120 868 R 0.3 0.1 0:00.03 top
  1 root 20 0 2828 1384 1192 S 0.0 0.1 0:01.92 init
  2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
  3 root RT 0 0 0 0 S 0.0 0.0 0:00.00 migration/0
  4 root 20 0 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/0
  5 root RT 0 0 0 0 S 0.0 0.0 0:00.00 watchdog/0
  6 root 20 0 0 0 0 S 0.0 0.0 0:00.26 events/0
  7 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuset
  8 root 20 0 0 0 0 S 0.0 0.0 0:00.00 khelper
  9 root 20 0 0 0 0 S 0.0 0.0 0:00.00 netns
 10 root 20 0 0 0 0 S 0.0 0.0 0:00.00 async/mgr
 11 root 20 0 0 0 0 S 0.0 0.0 0:00.00 pm
 12 root 20 0 0 0 0 S 0.0 0.0 0:00.01 sync_supers
 13 root 20 0 0 0 0 S 0.0 0.0 0:00.01 bdi-default
 14 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kintegrityd/0
 15 root 20 0 0 0 0 S 0.0 0.0 0:00.14 kblockd/0
 16 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kacpid
 17 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kacpi_notify
 18 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kacpi_hotplug
 19 root 20 0 0 0 0 S 0.0 0.0 0:02.38 ata/0

```

Figure 9.28: Output of the `top` Command

9.6.3 ps aux Command

The `ps aux` command is similar to the `ps ax` command. When a user executes the `ps aux` command, the owner of the process is also displayed with the process. Figure 9.29 displays the output of the `ps aux` command with the `less` pipe.

```

Access documents, folders and network places root@Marcus:~
File Edit View Search Terminal Help
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root 1 0.0 0.1 2828 1384 ?
root 2 0.0 0.0 0 0 ? S 07:37 0:01 /sbin/init
root 3 0.0 0.0 0 0 ? S 07:37 0:00 [kthreadd]
root 4 0.0 0.0 0 0 ? S 07:37 0:00 [migration/0]
root 5 0.0 0.0 0 0 ? S 07:37 0:00 [ksoftirqd/0]
root 6 0.0 0.0 0 0 ? S 07:37 0:00 [watchdog/0]
root 7 0.0 0.0 0 0 ? S 07:37 0:00 [events/0]
root 8 0.0 0.0 0 0 ? S 07:37 0:00 [cpuset]
root 9 0.0 0.0 0 0 ? S 07:37 0:00 [khelper]
root 10 0.0 0.0 0 0 ? S 07:37 0:00 [netns]
root 11 0.0 0.0 0 0 ? S 07:37 0:00 [pm]
root 12 0.0 0.0 0 0 ? S 07:37 0:00 [sync_supers]
root 13 0.0 0.0 0 0 ? S 07:37 0:00 [bdi-default]
root 14 0.0 0.0 0 0 ? S 07:37 0:00 [kintegrityd/0]
root 15 0.0 0.0 0 0 ? S 07:37 0:00 [kblockd/0]
root 16 0.0 0.0 0 0 ? S 07:37 0:00 [kacpid]
root 17 0.0 0.0 0 0 ? S 07:37 0:00 [kacpi_notify]
root 18 0.0 0.0 0 0 ? S 07:37 0:00 [kacpi_hotplug]
root 19 0.0 0.0 0 0 ? S 07:37 0:02 [ata/0]
root 20 0.0 0.0 0 0 ? S 07:37 0:00 [ata_aux]
root 21 0.0 0.0 0 0 ? S 07:37 0:00 [ksuspend_usbd]
root 22 0.0 0.0 0 0 ? S 07:37 0:00 [khubd]
root 23 0.0 0.0 0 0 ? S 07:37 0:00 [kseriod]
root 25 0.0 0.0 0 0 ? S 07:37 0:00 [khungtaskd]
root 26 0.0 0.0 0 0 ? S 07:37 0:00 [kswapd0]
root 27 0.0 0.0 0 0 ? SN 07:37 0:00 [ksmd]
root 28 0.0 0.0 0 0 ? S 07:37 0:00 [aio/0]
:
```

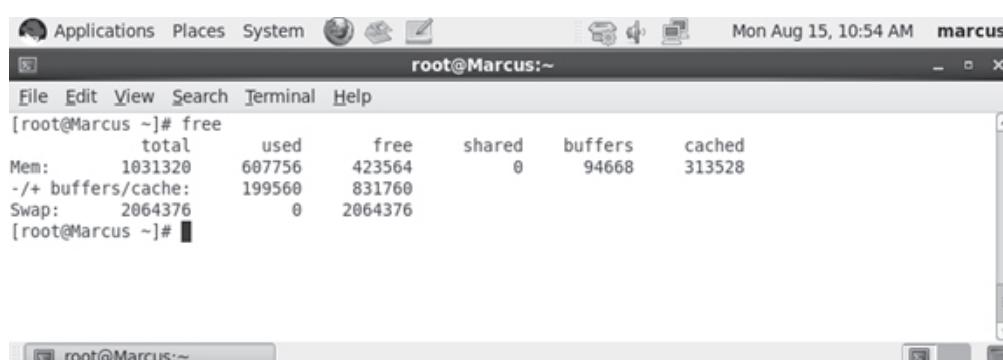
Figure 9.29: Output of the `ps aux` Command with the Less Pipe

9.7 Memory Usage

A user can view the memory usage using the `free` command. This command displays memory information that is as follows:

- Total physical memory
- Swap space
- Amount of memory used
- Free memory
- Shared memory
- Kernel buffers
- Cached

Figure 9.30 displays the output of the `free` command.



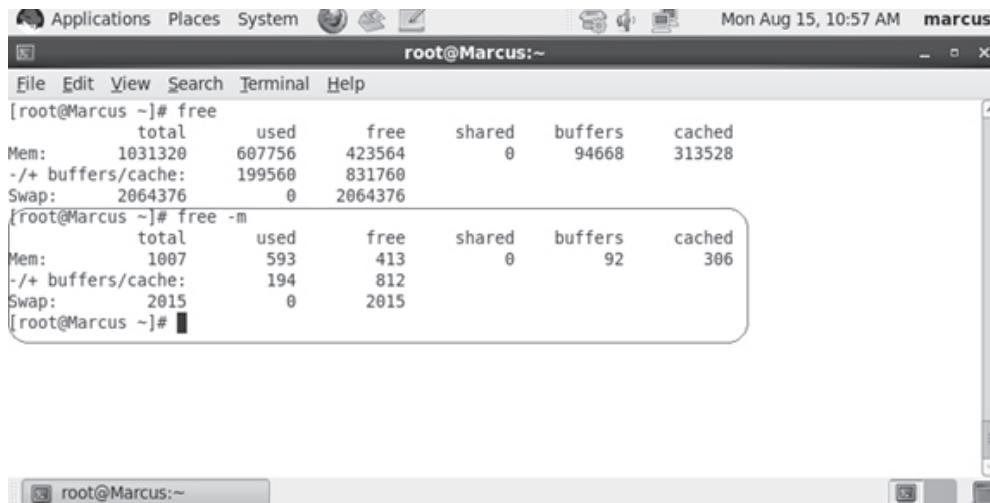
The screenshot shows a terminal window titled "root@Marcus:~". The window contains the output of the `free` command. The output is as follows:

```
[root@Marcus ~]# free
total        used        free      shared  buffers   cached
Mem:    1031320     607756     423564          0     94668   313528
-/+ buffers/cache:  199560     831760
Swap:   2064376          0    2064376
[root@Marcus ~]#
```

Figure 9.30: Output of the `free` Command

A user can use the `-m` parameter along with the `free` command to view the information in megabytes.

Figure 9.31 displays the output of the **free -m** command.



```
[root@Marcus ~]# free
total used free shared buffers cached
Mem: 1031320 607756 423564 0 94668 313528
-/+ buffers/cache: 199560 831760
Swap: 2064376 0 2064376
[root@Marcus ~]# free -m
total used free shared buffers cached
Mem: 1007 593 413 0 92 306
-/+ buffers/cache: 194 812
Swap: 2015 0 2015
[root@Marcus ~]#
```

Figure 9.31: Output of the **free -m** Command

In addition to using the **free** command, a user can use the graphical tool, which is System Monitor. The System Monitor can be invoked from **Applications → System Tools → System Monitor**. Figure 9.32 displays the System Monitor.

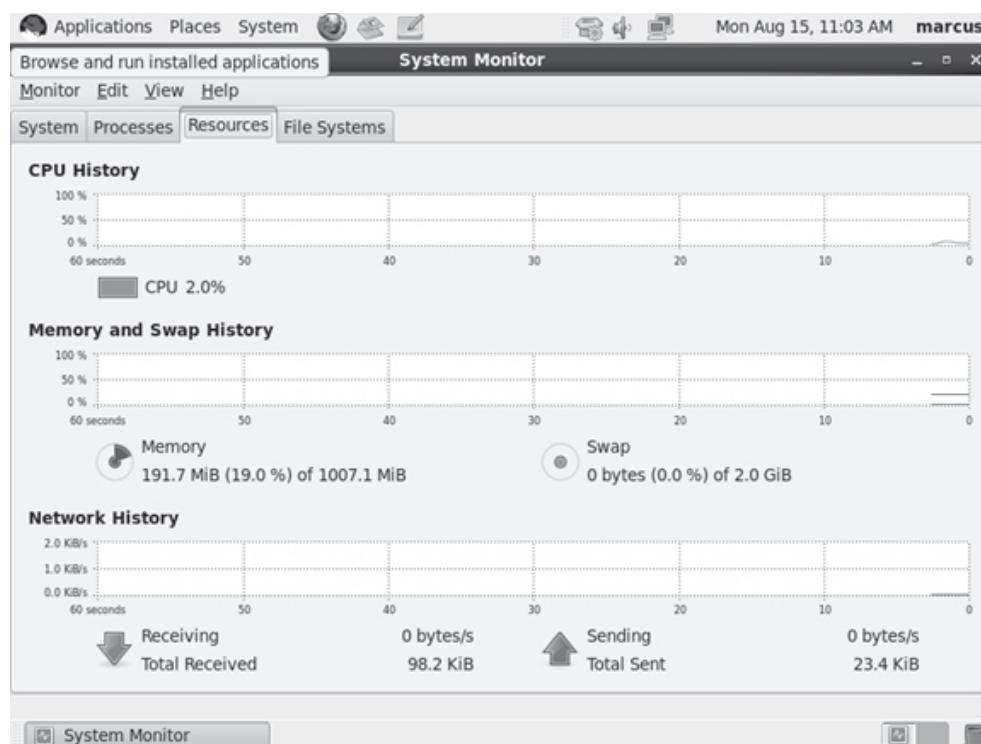


Figure 9.32: System Monitor

The Resources tab displays the utilization information of CPU, memory and swap and network. It displays the total and used amount of memory, CPU and network.

9.8 Automatic Bug Reporting Tool

Automatic Bug Reporting Tool (ABRT) is a daemon that is responsible for reporting application crashes. It is designed to run in the background but as soon as an application error is encountered, it comes in the foreground and reports the crash.

ABRT reports the crash data to its relevant issue tracker. For example, the crash data reports to the Red Hat Technical Support (RHTSupport). There are multiple trackers that accept crash data reports. The crashes that are reported can be managed by a user who can review, report or delete the crash data. It uses various plug-ins that are designed to report crash data to the relevant plug-ins.

The following components are part of the ABRT package:

- **Abrtd** - Is the system service.
- **Abrt-applet** - Runs in the user's Notification Area.
- **Abrt-gui** - Is the GUI application. It is responsible for showing crash data, which a user can edit, delete or report.
- **Abrt-cli** - Is similar to the abrt-gui application, which runs in the GUI environment. The abrt-cli application runs in the command line.

The ABRTGUI application can be invoked from **Applications → System Tools → Automatic Bug Reporting Tool**. Figure 9.33 displays the graphical interface of ABRT.

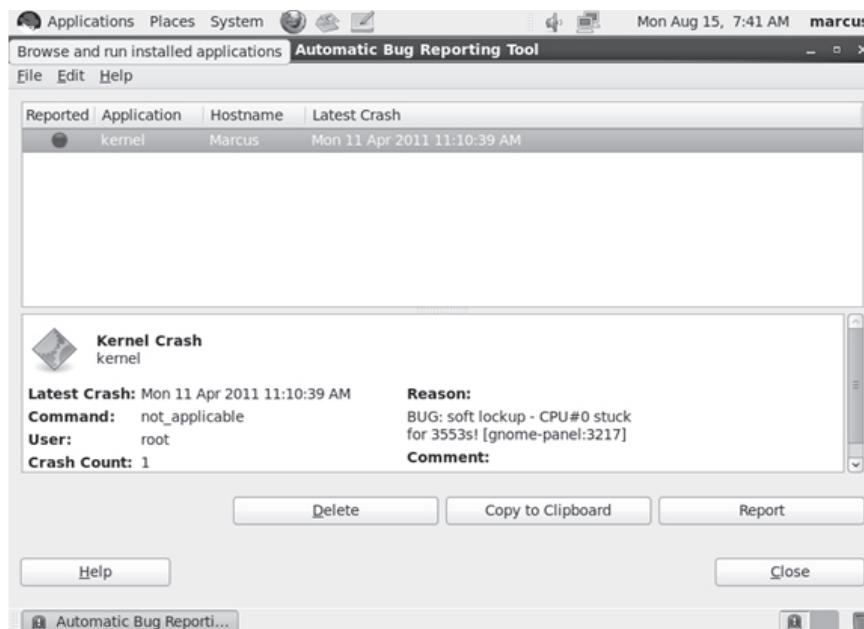


Figure 9.33: ABRT Graphical Interface

Users can add a number of plug-ins and add-ons to ABRT. Users can use the **yum** command to view all the available ABRT packages.

Following is the command that provides the details on the available ABRT packages:

```
yum list all | grep abrt
```

Figure 9.34 displays the available ABRT packages.

```
[marcus@Marcus Desktop]$ yum list all | grep abrt
*Note* Red Hat Network repositories are not listed below. You must run this command as root to access RHN repositories.
abrt.i686 1.1.13-4.el6 @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0
abrt-addon-ccpp.i686 1.1.13-4.el6 @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0
abrt-addon-kerneloops.i686 1.1.13-4.el6 @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0
abrt-addon-python.i686 1.1.13-4.el6 @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0
abrt-cli.i686 1.1.13-4.el6 @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0
abrt-desktop.i686 1.1.13-4.el6 @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0
abrt-gui.i686 1.1.13-4.el6 @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0
abrt-libs.i686 1.1.13-4.el6 @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0
abrt-plugin-logger.i686 1.1.13-4.el6 @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0
abrt-plugin-rhtsupport.i686 1.1.13-4.el6 @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0
abrt-plugin-sosreport.i686 1.1.13-4.el6 @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0
[marcus@Marcus Desktop]$
```

Figure 9.34: Available ABRT Packages

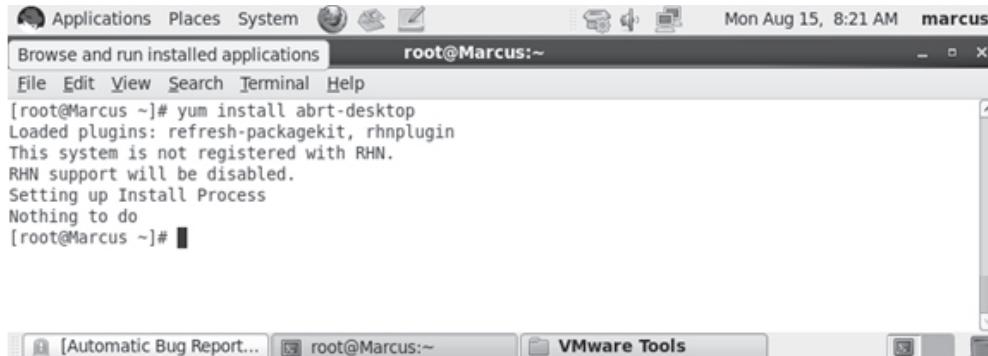
9.8.1 Installing and Running ABRT

ABRT is installed by default when RHEL is installed. To install ABRT, the **abrt-desktop** package must be installed on the RHEL system.

Following is the command that can be used to install ABRT:

```
yum install abrt-desktop
```

Figure 9.35 displays the execution of the command to Install ABRT.

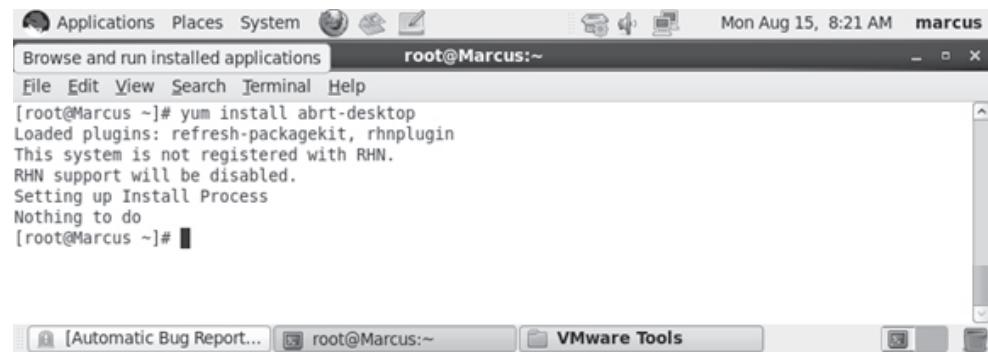


```
Applications Places System root@Marcus:~ Mon Aug 15, 8:21 AM marcus
Browse and run installed applications
File Edit View Search Terminal Help
[root@Marcus ~]# yum install abrt-desktop
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
Setting up Install Process
Nothing to do
[root@Marcus ~]#
```

Figure 9.35: Installing ABRT

If the package is not installed, the `yum` command installs the package. However, if the package is installed, the `yum` command does not re-install the package. It shows a message '**Nothing to do**'.

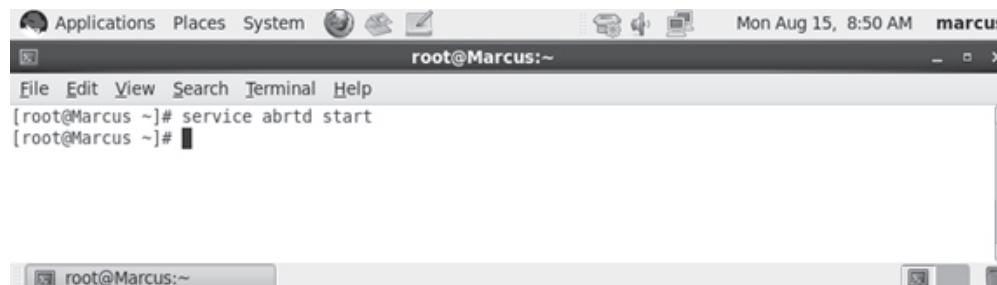
ABRT starts when the RHEL boots up. Users can verify the status of ABRT by executing the command as shown in Figure 9.36.



```
Applications Places System root@Marcus:~ Mon Aug 15, 8:21 AM marcus
Browse and run installed applications
File Edit View Search Terminal Help
[root@Marcus ~]# yum install abrt-desktop
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
Setting up Install Process
Nothing to do
[root@Marcus ~]#
```

Figure 9.36: ABRT Status

To start `abrtd` service, execute the following command `service abrtd start` as shown in Figure 9.37:

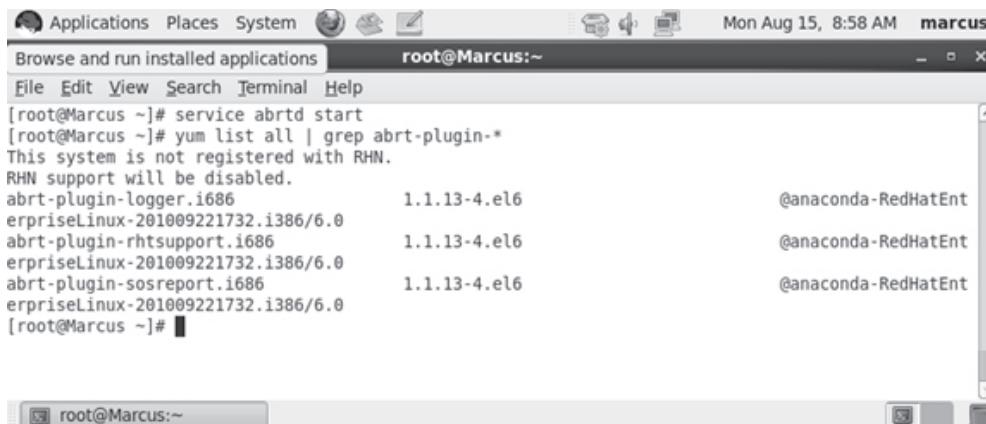


```
Applications Places System root@Marcus:~ Mon Aug 15, 8:50 AM marcus
File Edit View Search Terminal Help
[root@Marcus ~]# service abrtd start
[root@Marcus ~]#
```

Figure 9.37: Starting the abrtd Service

9.8.2 ABRT Plugins

ABRT uses a number of plug-ins that enhance its functionality. To view the available plug-ins, the following command must be executed as shown in Figure 9.38:



```
[root@Marcus ~]# service abrtd start
[root@Marcus ~]# yum list all | grep abrt-plugin-*
This system is not registered with RHN.
RHN support will be disabled.
abrt-plugin-logger.i686          1.1.13-4.el6           @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0
abrt-plugin-rhtsupport.i686      1.1.13-4.el6           @anaconda-RedHatEnt
enterpriseLinux-201009221732.i386/6.0
abrt-plugin-sosreport.i686       1.1.13-4.el6           @anaconda-RedHatEnt
enterpriseLinux-201009221732.i386/6.0
[root@Marcus ~]#
```

Figure 9.38: Available Plugins

These plug-ins are essentially divided into two categories. These categories are as follows:

- **Analyzer Plug-ins** - Helps to analyze the specific crash information. An example is Kernel loop plug-in that analyses the kernel crash only. These plug-ins are stored in the **/etc/abrt/plugins/** directory.
- **Reporter Plug-ins** - Helps to gather the information that is acquired by the analyzer plug-ins. The information is then provided as specific output. The Reporter plug-ins are configurable and each plug-in has its own configuration file. These plug-ins are stored in the **/etc/abrt/plugins/** directory. An example of this plug-in is RHT Support plug-in that crashes directly to the Red Hat Technical Support system.

9.8.3 Configuration File

A user can configure the Reporter plugins using the GUI or the ABRT. The user must invoke ABRT and click the Edit menu, and then select Plugins to invoke the Plugins dialog box.

Figure 9.39 displays the Plugins dialog box.

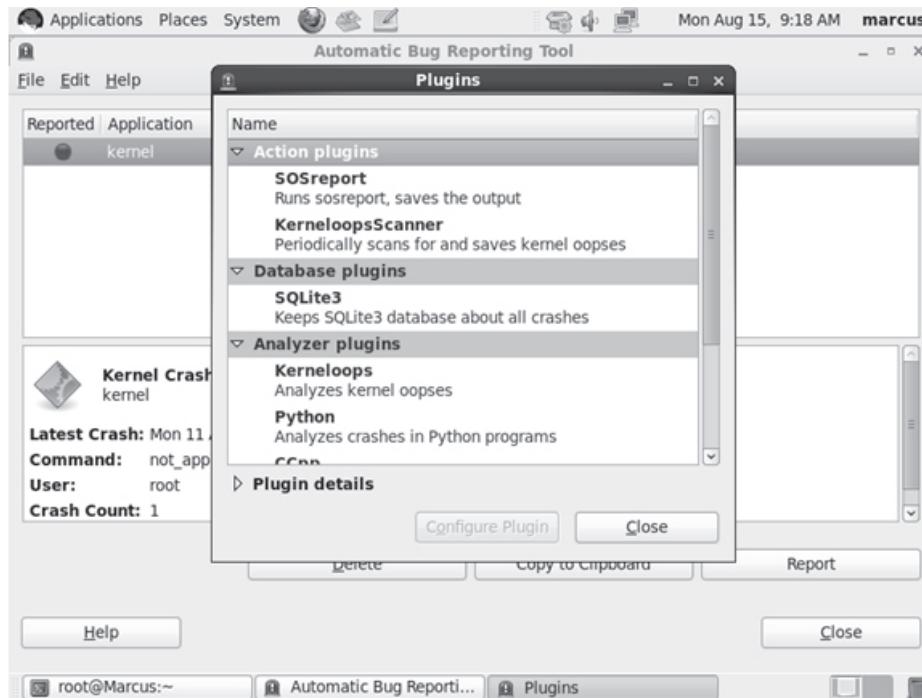


Figure 9.39: Plugins Dialog Box

To configure the plugin in the Plugins dialog box, the user must select a plugin in the Reporter plugins section, and then click Configure Plugin. The Logger dialog box is displayed in Figure 9.40.

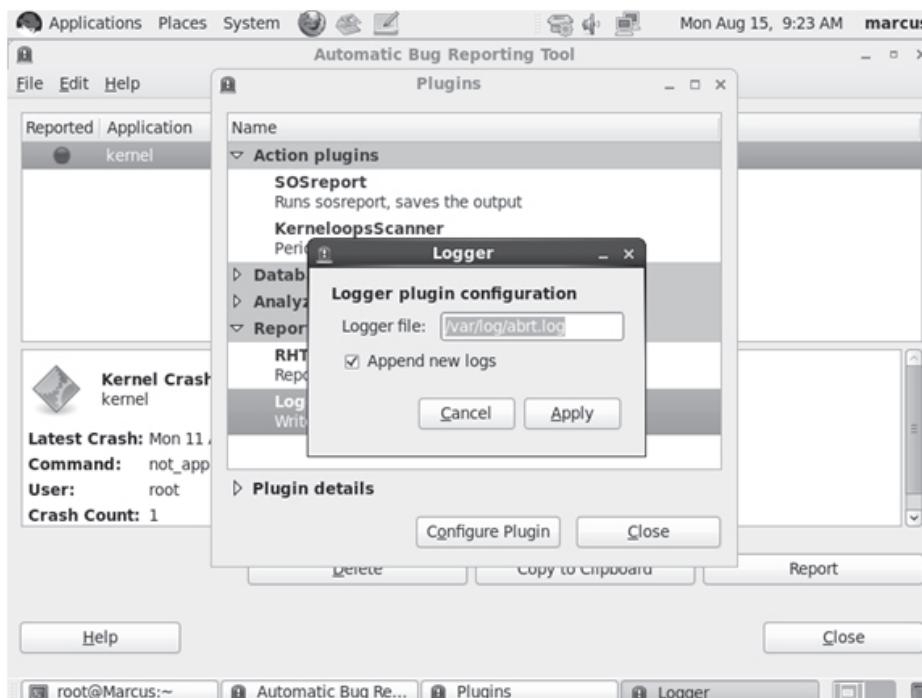


Figure 9.40: Logger Dialog Box

Users can view the configuration in this dialog box. The configuration files that are stored in the /etc/abrt/plugins/ directory are text files that can be used as global settings for the plugins. The user specific settings are stored in the Gnome key ring. Alternatively, the user specific settings can also be stored in a text file that is stored in the \$HOME/.abrt/*.config file. This file is used by the abrt-cli tool.

It is important to note that the user specific settings are configured only in the GUI environment. These settings are readable by the owner of \$HOME.

9.8.4 Backtraces

Each crash has a specific reason. The system administrators or the application developers must trace the information to figure out the reason for crash. Backtraces help the system administrators or the application developers in finding the information about the crash that has occurred with a program. ABRT, by default, is configured to generate backtraces for the crashes that occur.

Specific parameters are configured in the /etc/abrt/plugins/CCpp.conf file. Figure 9.41 displays the sample of CCpp.conf file that has backtraces enabled.



```
[root@Marcus ~]# cat /etc/abrt/plugins/CCpp.conf
# Configuration file for CCpp hook and plugin
Enabled = yes

# If you also want to dump file named "core"
# in crashed process' current dir, set to "yes"
MakeCompatCore = yes

# Do you want a copy of crashed binary be saved?
# (useful, for example, when _deleted_ binary_ segfaults)
SaveBinaryImage = no

# Generate backtrace
Backtrace = yes

# Generate backtrace for crashes uploaded from remote machines.
# Note that for reliable backtrace generation, your local machine
# needs to have the crashed executable and all libraries it uses,
# and they need to be the same versions as on remote machines.
# If you cannot ensure that, it's better to set this option to "no"
BacktraceRemotes = no

# Generate memory map too (IGNORED FOR NOW)
MemoryMap = no

# How to get debuginfo: install, mount
## install - download and install debuginfo packages
## mount - mount fedora NFS with debug info
## (IGNORED FOR NOW)
```

Figure 9.41: Sample of CCpp.conf File

9.9 Check Your Progress

1. Which package must be installed to execute the **batch** command?

(A)	at	(C)	anacron
(B)	batch	(D)	cron

2. Which file contains the configuration for the anacron daemon?

(A)	anacrontab.conf	(C)	cron
(B)	anacrontab	(D)	crontab.conf

3. Which directory is used to start the crond service?

(A)	/etc/	(C)	/bin/
(B)	/var/	(D)	/sbin/

4. Which parameter is required if a user has to start the service again without stopping it?

(A)	stop	(C)	status
(B)	restart	(D)	start

5. Which command is used for a specific job at a scheduled time?

(A)	cron	(C)	at
(B)	anacron	(D)	crond

6. Which directory under the /etc directory contains the configuration files?

(A)	sysconfig	(C)	sys-config
(B)	systemconfig	(D)	system-config

7. Which is the correct command to get the package details from a configuration file?

(A)	<i>yum provides</i>	(C)	<i>yum info</i>
(B)	<i>yum details</i>	(D)	<i>yum filename</i>

8. Which of the following is the directory under \etc\sysconfig?

(A)	networking-scripts	(C)	samba
(B)	yum	(D)	http

9. Which is the following is the correct path for the file that can be used for getting complete information on sysconfig directory?

(A)	/usr/doc/sysconfig.txt	(C)	/usr/doc/initscripts-4.48/ sysconfig.txt
(B)	/usr/var/sysconfig.txt	(D)	/usr/doc/initscripts/ sysconfig.txt

10. Which of the following is a directory in sysconfig directory that is typically installed with RHEL installation?

(A)	Network-files	(C)	Network-connections
(B)	Network-scripts	(D)	Network-connect

11. Where are the configuration files stored for the ABRT tool?

(A)	/etc/abrt/plugins/	(C)	/usr/abrt/plugins
(B)	/var/abrt/plugins	(D)	/tmp/abrt/plugins

12. Where are the user specific settings for ABRT stored?

(A)	\$HOME / .abrt/*.config	(C)	\$HOME /*.config
(B)	\$HOME /logs	(D)	\$HOME /config

13. Which command allows a user to view the real-time information for processes?

(A)	Top	(C)	ps -ax
(B)	ps	(D)	free

14. Which command is used for viewing memory utilization?

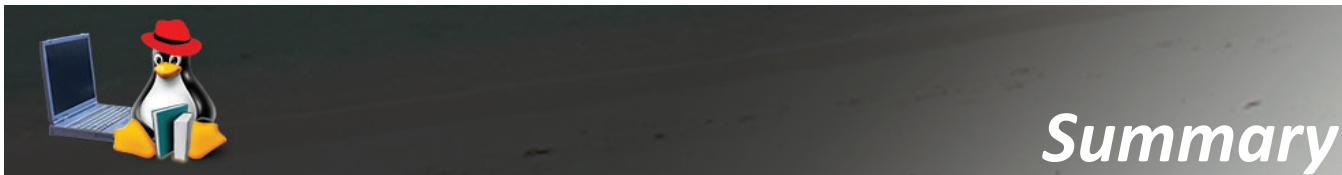
(A)	Top	(C)	ps -ax
(B)	ps	(D)	free

15. Which parameter is used with `yum install` command for installing ABRT on the command line?

(A)	abrt-desktop	(C)	desktop
(B)	abrtd	(D)	abrtd desktop

9.9.1 Answers

1.	A
2.	B
3.	D
4.	A
5.	C
6.	A
7.	A
8.	A
9.	C
10.	B
11.	A
12.	A
13.	A
14.	D
15.	A



- In RHEL, jobs can either run manually or automatically at a scheduled time. A number of commands can be used to schedule tasks. These commands are cron, anacron, at, and batch. cron is a daemon that is used to schedule jobs. The cron command is linked with the crond service. cron assumes that the systems are always in running state. The anacron command runs a job only once a day. If the system is turned OFF, the anacron daemon runs the job next time when the system is running.
- The at command is used to run a specific job at a scheduled time. The batch command is similar to the at command. However, the batch command executes one-time tasks only when the load is less than 0.8% on the server.
- The sysconfig directory stores the configuration files that are responsible for controlling the system configuration. There are two key components of this directory namely, files and directories. The number of files and directories depend on the programs a user installs on a RHEL system. A user can use the yum provides command to get the details of a specific program using its configuration file.
- Some of the system resources a user should know about are CPU, memory and hard disk space. It is essential for appropriately utilizing the system resources. The system processes can be viewed using top, ps ax, and ps aux commands. Also, the free command is used for viewing memory utilization along with swap file utilization.
- The ABRT tool is used for tracking information on crashes that has occurred within a system. There are two types of plugins that are typically installed. These are Analyzers and Reporter plugins. Backtraces are used for providing more details about a crash. Backtraces are used by system administrators and developers.



Need
HELP
on a topic? = **FAQs**



www.onlinevarsity.com

Exercise - 1**Working with the vi editor**

Note - The **vi** editor, is a command-line-based editor. This lab requires vim-minimal RPM to be installed on the computer.

Step 1 - Log on to the RHEL workstation with the **username** and **password**.

Step 2 - Open the command line from **Applications → System Tools → Terminal**. The command-line terminal appears as shown in Figure 10.1.

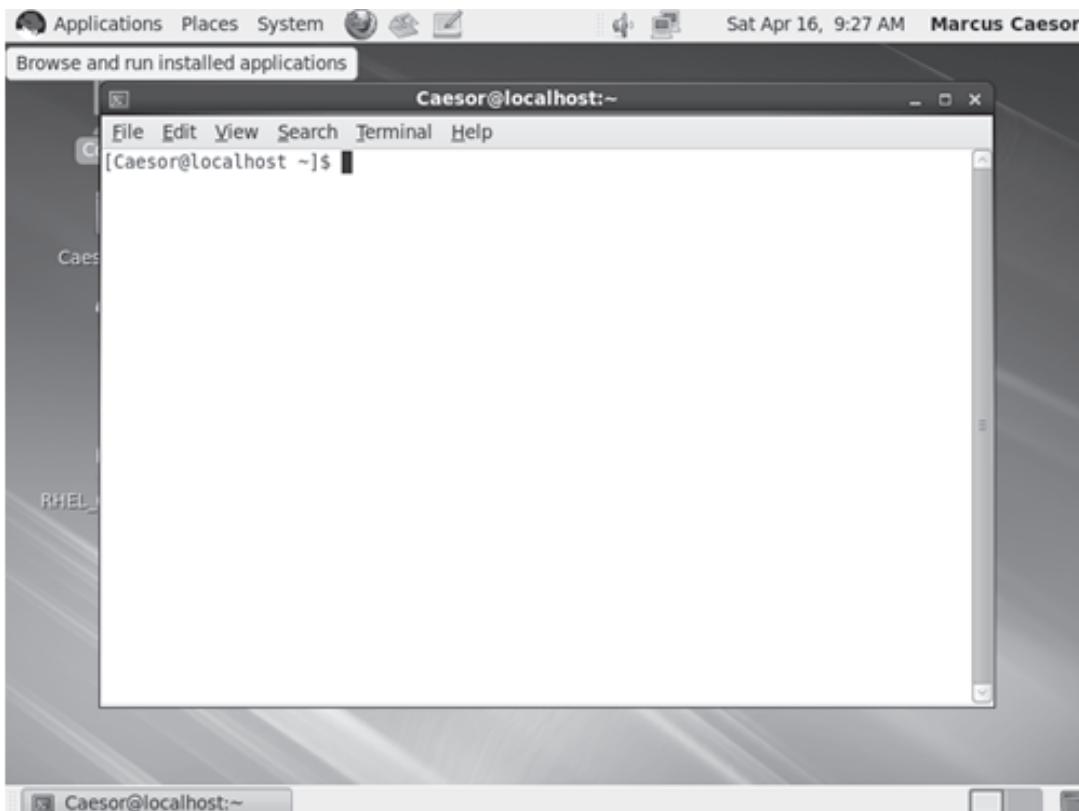
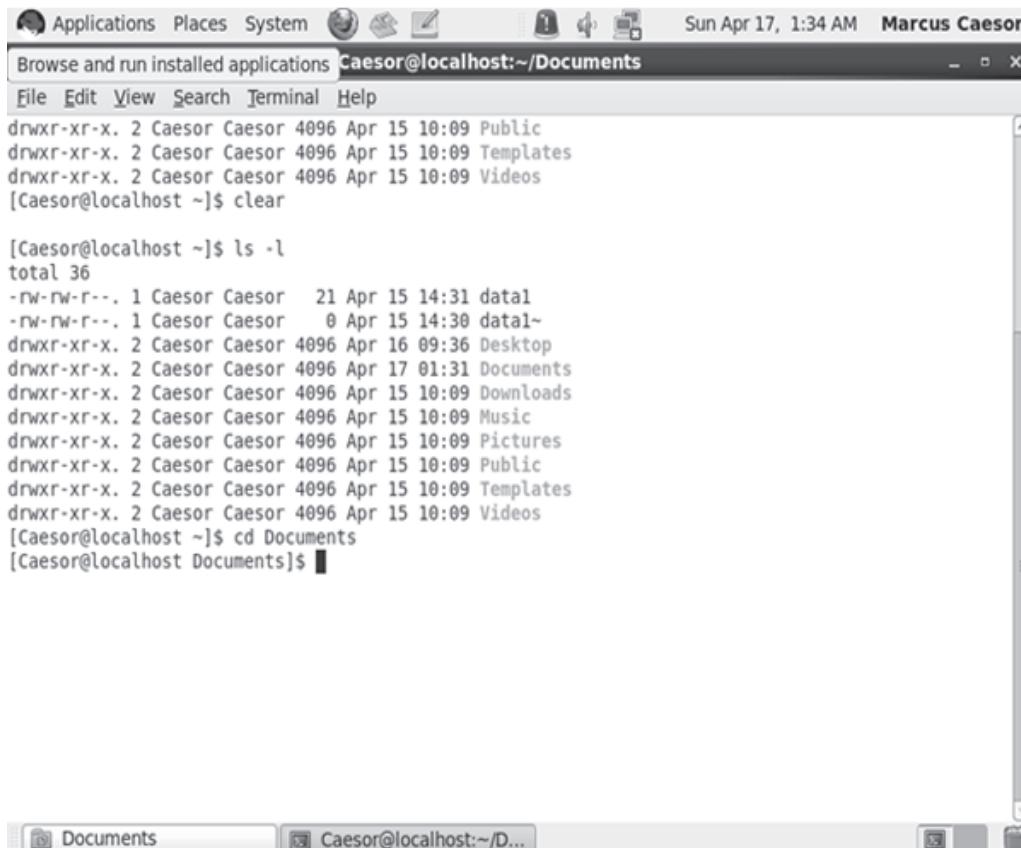


Figure 10.1: Command-Line Terminal

Note - Alternatively, right-click the Desktop and click the **Open in Terminal** option to open command-line terminal.

Step 3 - Type `ls -l` at the command prompt and press ENTER. The directory listing is displayed as shown in Figure 10.2.



The screenshot shows a terminal window titled "Caesor@localhost:~/Documents". The window contains a command-line interface where the user has run the "ls -l" command. The output of the command is a detailed listing of files and directories in the current directory. The listing includes file names, permissions (e.g., drwxr-xr-x), owner (Caesor), group (Caesor), size (e.g., 4096), modification date (e.g., Apr 15), and time (e.g., 10:09). Other commands shown in the terminal include "clear" and "cd Documents". The desktop environment visible behind the terminal window includes icons for Applications, Places, System, and a menu bar. The taskbar at the bottom shows the "Documents" application is active.

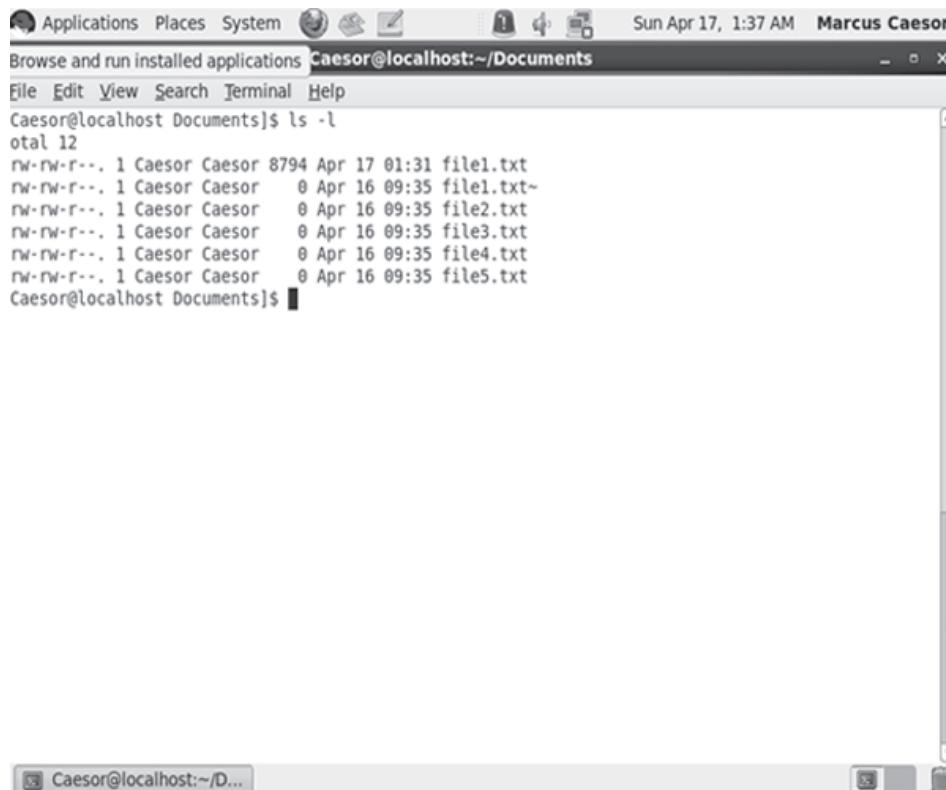
```
drwxr-xr-x. 2 Caesor Caesor 4096 Apr 15 10:09 Public
drwxr-xr-x. 2 Caesor Caesor 4096 Apr 15 10:09 Templates
drwxr-xr-x. 2 Caesor Caesor 4096 Apr 15 10:09 Videos
[Caesor@localhost ~]$ clear

[Caesor@localhost ~]$ ls -l
total 36
-rw-rw-r--. 1 Caesor Caesor 21 Apr 15 14:31 data1
-rw-rw-r--. 1 Caesor Caesor 0 Apr 15 14:30 data1~
drwxr-xr-x. 2 Caesor Caesor 4096 Apr 16 09:36 Desktop
drwxr-xr-x. 2 Caesor Caesor 4096 Apr 17 01:31 Documents
drwxr-xr-x. 2 Caesor Caesor 4096 Apr 15 10:09 Downloads
drwxr-xr-x. 2 Caesor Caesor 4096 Apr 15 10:09 Music
drwxr-xr-x. 2 Caesor Caesor 4096 Apr 15 10:09 Pictures
drwxr-xr-x. 2 Caesor Caesor 4096 Apr 15 10:09 Public
drwxr-xr-x. 2 Caesor Caesor 4096 Apr 15 10:09 Templates
drwxr-xr-x. 2 Caesor Caesor 4096 Apr 15 10:09 Videos
[Caesor@localhost ~]$ cd Documents
[Caesor@localhost Documents]$
```

Figure 10.2: Directory Listing

Step 4 - Execute `cd Documents` at the command prompt.

Step 5 - Execute `ls -l` at the command prompt. A detailed listing of all the files that are available in the Documents directory is displayed as shown in Figure 10.3.



The screenshot shows a terminal window titled "Caesor@localhost:~/Documents". The window contains the following text:

```
Applications Places System Sun Apr 17, 1:37 AM Marcus Caesor
Browse and run installed applications Caesor@localhost:~/Documents
File Edit View Search Terminal Help
Caesor@localhost Documents]$ ls -l
total 12
-rw-rw-r--. 1 Caesor Caesor 8794 Apr 17 01:31 file1.txt
-rw-rw-r--. 1 Caesor Caesor 0 Apr 16 09:35 file1.txt~
-rw-rw-r--. 1 Caesor Caesor 0 Apr 16 09:35 file2.txt
-rw-rw-r--. 1 Caesor Caesor 0 Apr 16 09:35 file3.txt
-rw-rw-r--. 1 Caesor Caesor 0 Apr 16 09:35 file4.txt
-rw-rw-r--. 1 Caesor Caesor 0 Apr 16 09:35 file5.txt
Caesor@localhost Documents]$
```

Figure 10.3: Detailed Listing of Files

Note - The `file1.txt~` is a backup file, which is temporary. This directory also contains five more files as shown in Figure 10.3.

Step 6 - Type `vi file1.txt` and press ENTER. The `file1.txt` file is opened.

Step 7 - The cursor would be at the first character of the first word in the document. To modify the file, the user must start the Insert mode. Press the **i** key. The document switches to the Insert mode as shown in Figure 10.4.

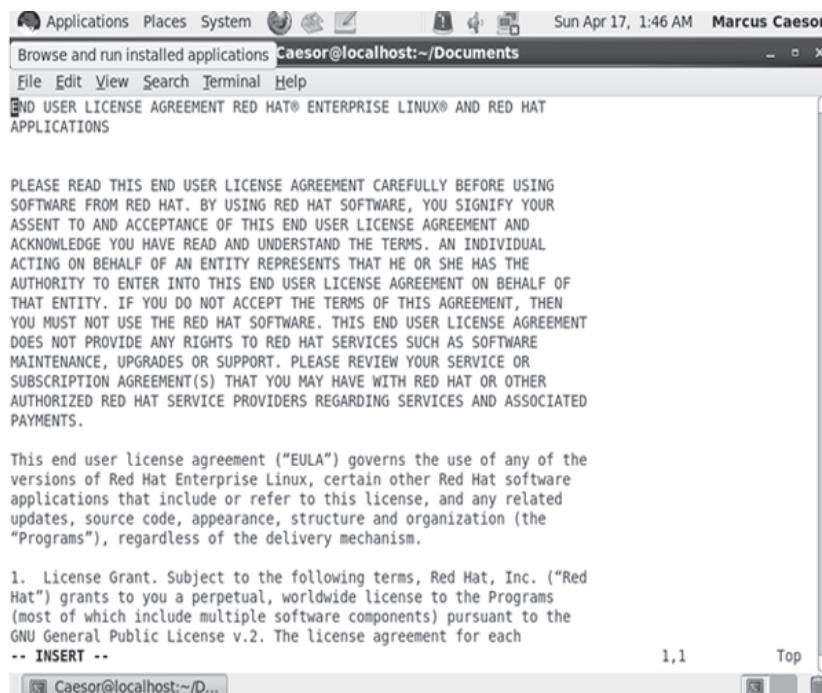


Figure 10.4: Insert Mode

Step 8 - Type the name, and then press **SPACE BAR** in the Insert mode. This adds the name along with a blank space at the start of the document. (Refer to Figure 10.5)

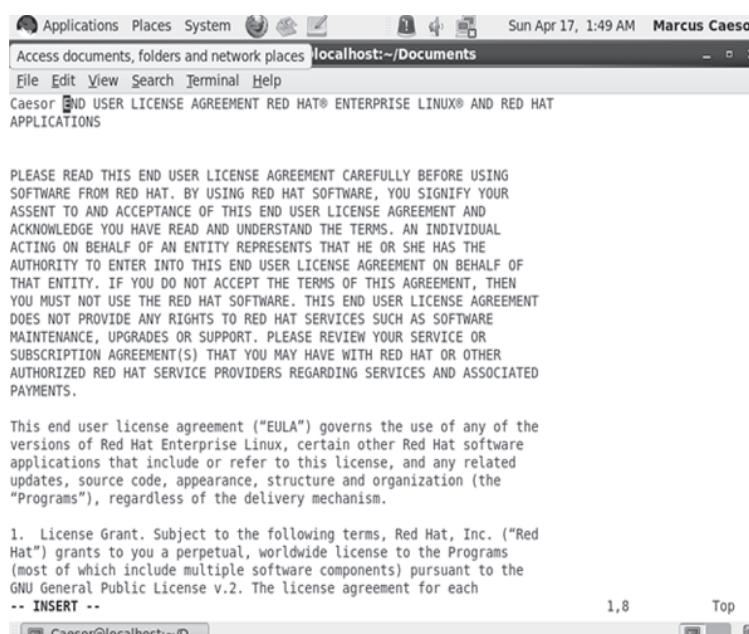
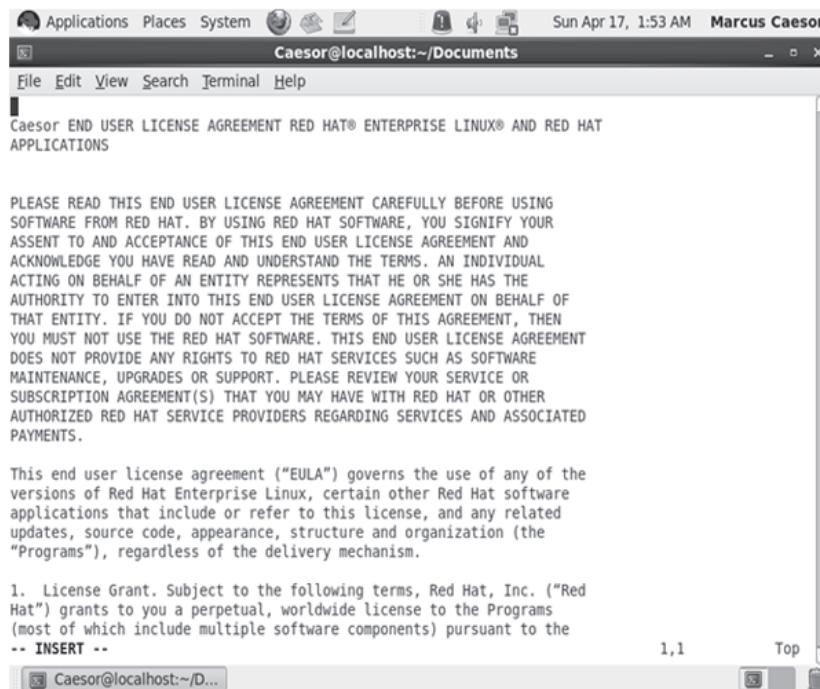


Figure 10.5: Add Name with a Blank Space

Step 9 - Press **ESC** to exit the Insert mode, and then press the **SHIFT+O** key combination to insert a blank line before the first line. A new blank line is inserted before the first line as shown in Figure 10.6.



```

Applications Places System Sun Apr 17, 1:53 AM Marcus Caesor
Caesor@localhost:~/Documents
File Edit View Search Terminal Help
Caesor END USER LICENSE AGREEMENT RED HAT® ENTERPRISE LINUX® AND RED HAT
APPLICATIONS

PLEASE READ THIS END USER LICENSE AGREEMENT CAREFULLY BEFORE USING
SOFTWARE FROM RED HAT. BY USING RED HAT SOFTWARE, YOU SIGNIFY YOUR
ASSENT TO AND ACCEPTANCE OF THIS END USER LICENSE AGREEMENT AND
ACKNOWLEDGE YOU HAVE READ AND UNDERSTAND THE TERMS. AN INDIVIDUAL
ACTING ON BEHALF OF AN ENTITY REPRESENTS THAT HE OR SHE HAS THE
AUTHORITY TO ENTER INTO THIS END USER LICENSE AGREEMENT ON BEHALF OF
THAT ENTITY. IF YOU DO NOT ACCEPT THE TERMS OF THIS AGREEMENT, THEN
YOU MUST NOT USE THE RED HAT SOFTWARE. THIS END USER LICENSE AGREEMENT
DOES NOT PROVIDE ANY RIGHTS TO RED HAT SERVICES SUCH AS SOFTWARE
MAINTENANCE, UPGRADES OR SUPPORT. PLEASE REVIEW YOUR SERVICE OR
SUBSCRIPTION AGREEMENT(S) THAT YOU MAY HAVE WITH RED HAT OR OTHER
AUTHORIZED RED HAT SERVICE PROVIDERS REGARDING SERVICES AND ASSOCIATED
PAYMENTS.

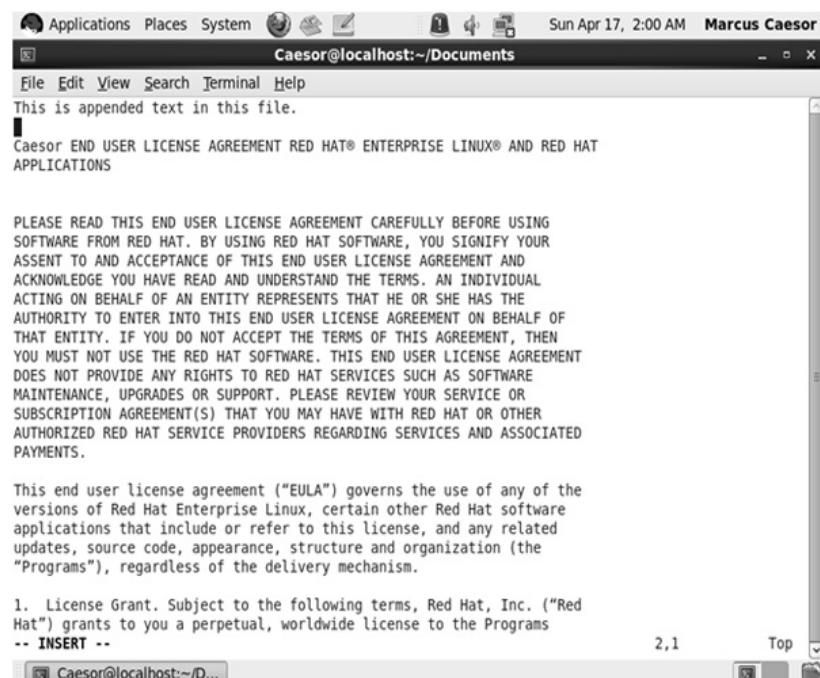
This end user license agreement ("EULA") governs the use of any of the
versions of Red Hat Enterprise Linux, certain other Red Hat software
applications that include or refer to this license, and any related
updates, source code, appearance, structure and organization (the
"Programs"), regardless of the delivery mechanism.

1. License Grant. Subject to the following terms, Red Hat, Inc. ("Red
Hat") grants to you a perpetual, worldwide license to the Programs
(most of which include multiple software components) pursuant to the
-- INSERT --
1,1 Top
Caesor@localhost:~/D...

```

Figure 10.6: Inserting a Blank Line in the Document

Step 10 - Type '**This is appended text in this file**', and then press **ENTER**. A new blank line is inserted after the appended text as shown in Figure 10.7.



```

Applications Places System Sun Apr 17, 2:00 AM Marcus Caesor
Caesor@localhost:~/Documents
File Edit View Search Terminal Help
This is appended text in this file.

Caesor END USER LICENSE AGREEMENT RED HAT® ENTERPRISE LINUX® AND RED HAT
APPLICATIONS

PLEASE READ THIS END USER LICENSE AGREEMENT CAREFULLY BEFORE USING
SOFTWARE FROM RED HAT. BY USING RED HAT SOFTWARE, YOU SIGNIFY YOUR
ASSENT TO AND ACCEPTANCE OF THIS END USER LICENSE AGREEMENT AND
ACKNOWLEDGE YOU HAVE READ AND UNDERSTAND THE TERMS. AN INDIVIDUAL
ACTING ON BEHALF OF AN ENTITY REPRESENTS THAT HE OR SHE HAS THE
AUTHORITY TO ENTER INTO THIS END USER LICENSE AGREEMENT ON BEHALF OF
THAT ENTITY. IF YOU DO NOT ACCEPT THE TERMS OF THIS AGREEMENT, THEN
YOU MUST NOT USE THE RED HAT SOFTWARE. THIS END USER LICENSE AGREEMENT
DOES NOT PROVIDE ANY RIGHTS TO RED HAT SERVICES SUCH AS SOFTWARE
MAINTENANCE, UPGRADES OR SUPPORT. PLEASE REVIEW YOUR SERVICE OR
SUBSCRIPTION AGREEMENT(S) THAT YOU MAY HAVE WITH RED HAT OR OTHER
AUTHORIZED RED HAT SERVICE PROVIDERS REGARDING SERVICES AND ASSOCIATED
PAYMENTS.

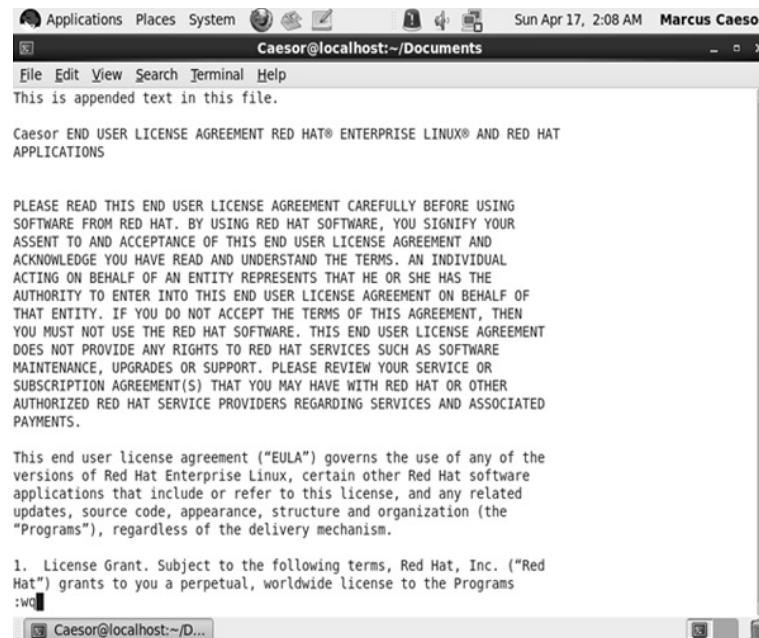
This end user license agreement ("EULA") governs the use of any of the
versions of Red Hat Enterprise Linux, certain other Red Hat software
applications that include or refer to this license, and any related
updates, source code, appearance, structure and organization (the
"Programs"), regardless of the delivery mechanism.

1. License Grant. Subject to the following terms, Red Hat, Inc. ("Red
Hat") grants to you a perpetual, worldwide license to the Programs
-- INSERT --
2,1 Top
Caesor@localhost:~/D...

```

Figure 10.7: Inserting New Blank Line

Step 11 - To save the file, press **ESC**, and then type **:wq**. The **:wq** characters appear at the bottom of the file as shown in Figure 10.8.



The screenshot shows a terminal window titled "Caesor@localhost:~/Documents". The window contains the following text:

```

Applications Places System Sun Apr 17, 2:08 AM Marcus Caesor
File Edit View Search Terminal Help
Caesor@localhost:~/Documents
This is appended text in this file.

Caeser END USER LICENSE AGREEMENT RED HAT® ENTERPRISE LINUX® AND RED HAT
APPLICATIONS

PLEASE READ THIS END USER LICENSE AGREEMENT CAREFULLY BEFORE USING
SOFTWARE FROM RED HAT. BY USING RED HAT SOFTWARE, YOU SIGNIFY YOUR
ASSENT TO AND ACCEPTANCE OF THIS END USER LICENSE AGREEMENT AND
ACKNOWLEDGE YOU HAVE READ AND UNDERSTAND THE TERMS. AN INDIVIDUAL
ACTING ON BEHALF OF AN ENTITY REPRESENTS THAT HE OR SHE HAS THE
AUTHORITY TO ENTER INTO THIS END USER LICENSE AGREEMENT ON BEHALF OF
THAT ENTITY. IF YOU DO NOT ACCEPT THE TERMS OF THIS AGREEMENT, THEN
YOU MUST NOT USE THE RED HAT SOFTWARE. THIS END USER LICENSE AGREEMENT
DOES NOT PROVIDE ANY RIGHTS TO RED HAT SERVICES SUCH AS SOFTWARE
MAINTENANCE, UPGRADES OR SUPPORT. PLEASE REVIEW YOUR SERVICE OR
SUBSCRIPTION AGREEMENT(S) THAT YOU MAY HAVE WITH RED HAT OR OTHER
AUTHORIZED RED HAT SERVICE PROVIDERS REGARDING SERVICES AND ASSOCIATED
PAYMENTS.

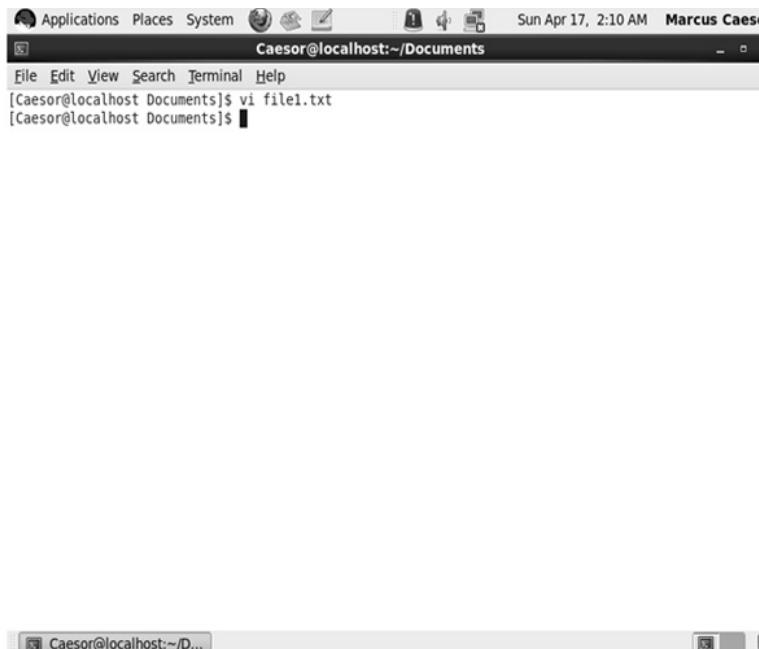
This end user license agreement ("EULA") governs the use of any of the
versions of Red Hat Enterprise Linux, certain other Red Hat software
applications that include or refer to this license, and any related
updates, source code, appearance, structure and organization (the
"Programs"), regardless of the delivery mechanism.

1. License Grant. Subject to the following terms, Red Hat, Inc. ("Red
Hat") grants to you a perpetual, worldwide license to the Programs
:wq

```

Figure 10.8: :wq Characters

Step 12 - Press **ENTER**. The file is now saved and then closed. The user now returns to the command prompt as shown in Figure 10.9.



The screenshot shows a terminal window titled "Caesor@localhost:~/Documents". The window contains the following text:

```

Applications Places System Sun Apr 17, 2:10 AM Marcus Caesor
File Edit View Search Terminal Help
[Caesor@localhost Documents]$ vi file1.txt
[Caesor@localhost Documents]$ 

```

Figure 10.9: Command Prompt

Note - The user can press **ESC** to exit the Insert mode, and then type: **q!** to not save the changes to the file.

Exercise - 2

Using the **more** command

To use the **more** commands, the user should perform the following steps.

Step 1 - At the command prompt, type **more file1.txt** and press **ENTER**. The contents of the **file1.txt** file are displayed as shown in Figure 10.10.

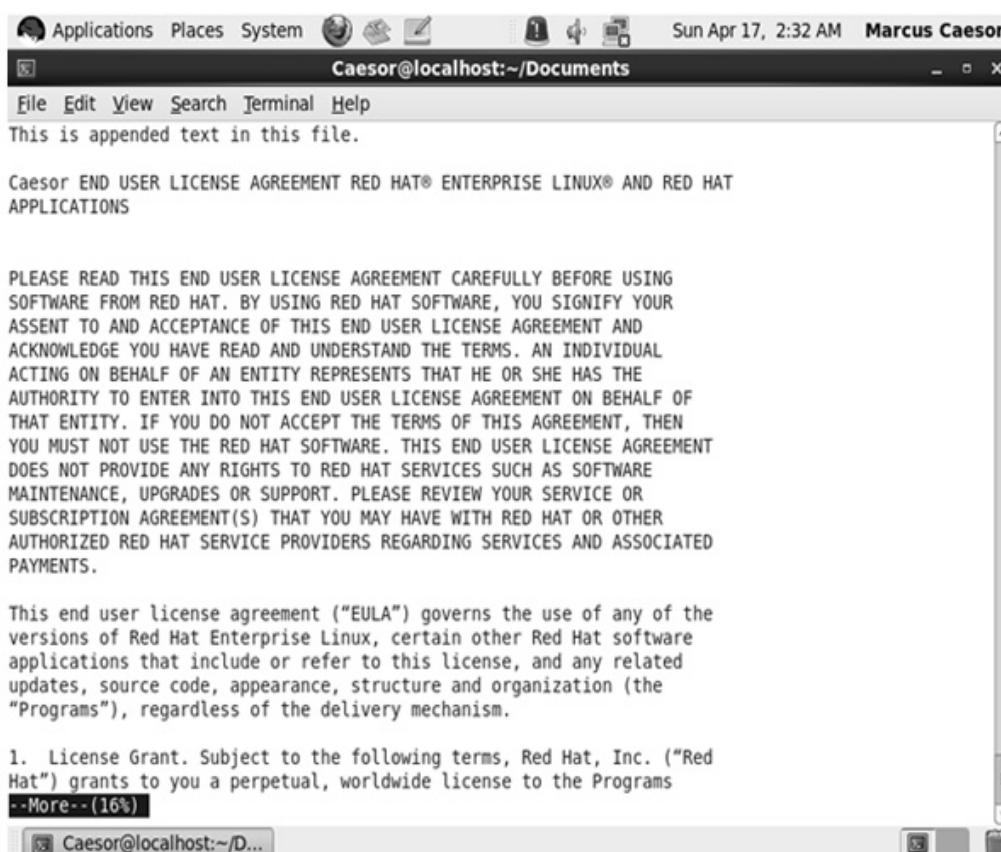


Figure 10.10: Contents of **file1.txt** File

Step 2 - Press **SPACE BAR** twice and notice the percentage (16%) shown at the bottom of the screen. It should significantly change the percentage (54%) as shown in Figure 10.11.

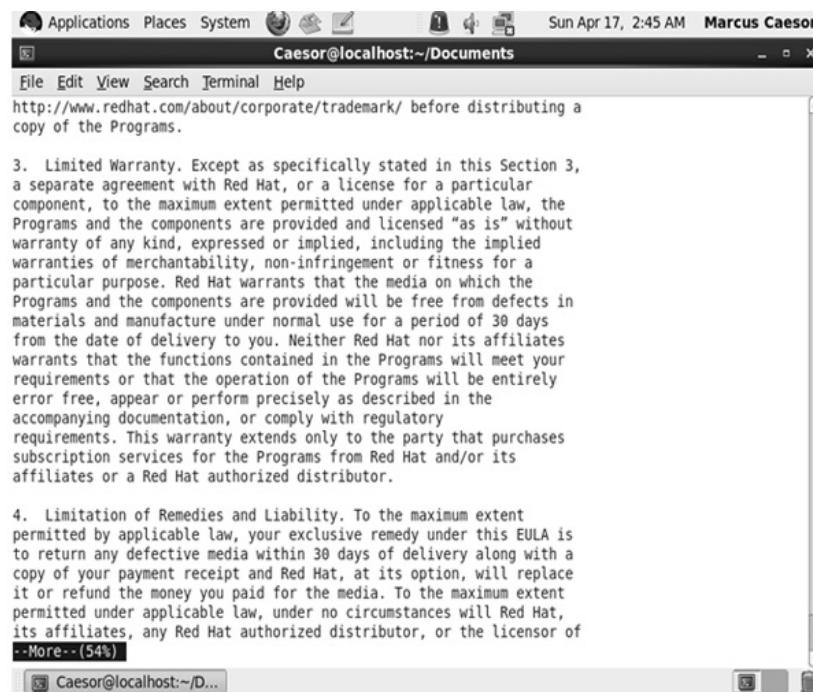


Figure 10.11: Significant Change in the Percentage

Step 3 - Press **ENTER** twice. The changes in the percentage are very minimal as shown in Figure 10.12.

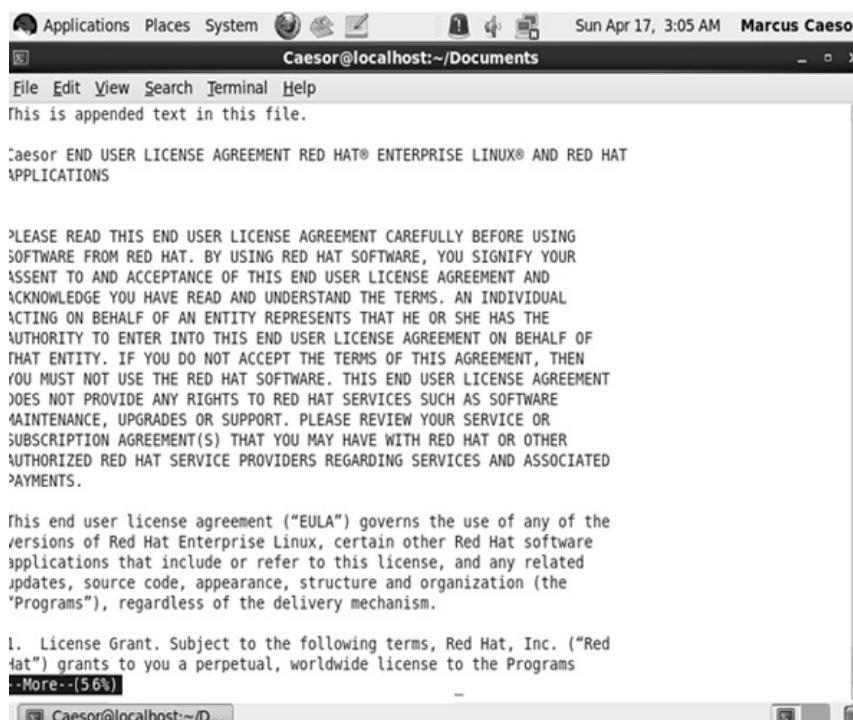


Figure 10.12: Minimal Change in the Percentage

Step 4 - In the file1.txt file, press **CTRL+Z** to exit the **more** command. The command prompt appears.

Step 5 - Type the following command at the command prompt:

```
more -d file1.txt
```

Step 6 - Press **ENTER**. The file1.txt file is opened as shown in Figure 10.13.

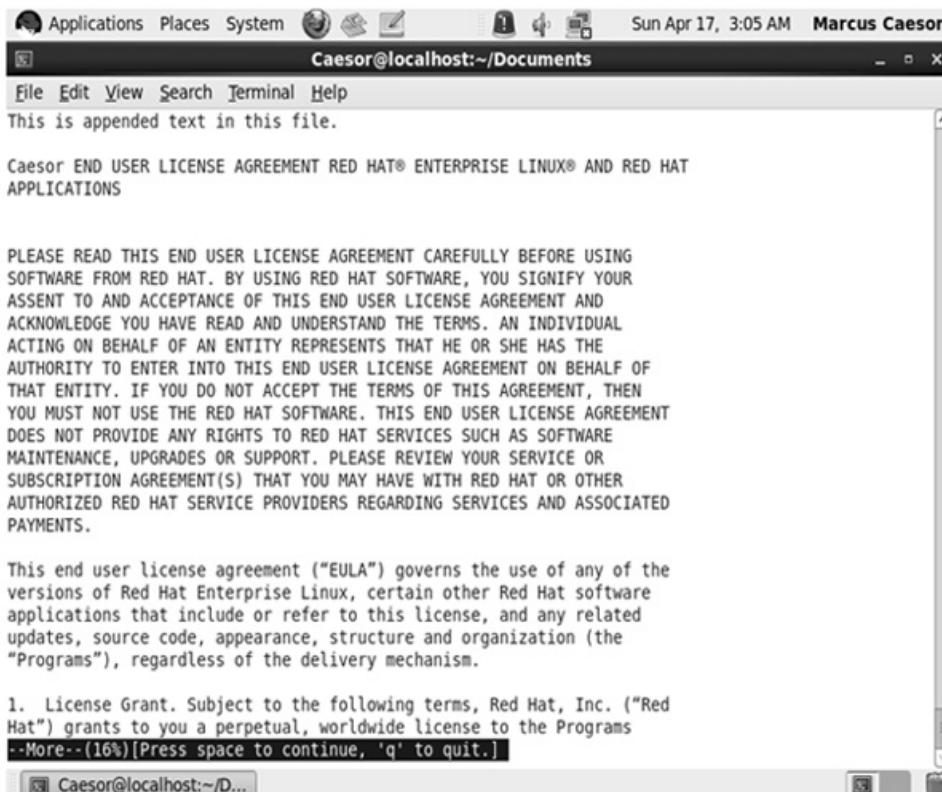


Figure 10.13: file1.txt File

Step 7 - Press the **q** key to exit the **more** command and return to the command prompt.

Note - The **less** command is very similar to the **more** command. The key differences of the **less** command are as follows:

- Allows backward and forward navigation
- Does not read the entire file before displaying the contents on the screen

Exercise - 3

Using the **cat** and **file** command

There can be a situation when the user has finished using the **more** command. Before using other commands, the user should clear the screen.

To clear the screen, perform the following steps:

Step 1 - Type `clear` and press **ENTER**.

Step 2 - Type `cat -n file1.txt` and press **ENTER**. Each line in the `file1.txt` file is marked with the corresponding line number as shown in Figure 10.14.

```

Applications Places System Sun Apr 17, 3:33 AM Marcus Caesor
Caesor@localhost:~/Documents
e Edit View Search Terminal Help
123
124 6. Third Party Programs. Red Hat may distribute third party software
125 programs with the Programs that are not part of the Programs. These
126 third party programs are not required to run the Programs, are
127 provided as a convenience to you, and are subject to their own license
128 terms. The license terms either accompany the third party software
129 programs or can be viewed at
130 http://www.redhat.com/licenses/thirdparty/eula.html. If you do not
131 agree to abide by the applicable license terms for the third party
132 software programs, then you may not install them. If you wish to
133 install the third party software programs on more than one system or
134 transfer the third party software programs to another party, then you
135 must contact the licensor of the applicable third party software
136 programs.
137
138 7. General. If any provision of this EULA is held to be unenforceable,
139 the enforceability of the remaining provisions shall not be
140 affected. Any claim, controversy or dispute arising under or relating
141 to this EULA shall be governed by the laws of the State of New York
142 and of the United States, without regard to any conflict of laws
143 provisions. The rights and obligations of the parties to this EULA
144 shall not be governed by the United Nations Convention on the
145 International Sale of Goods.
146
147 Copyright © 2010 Red Hat, Inc. All rights reserved. "Red Hat" and the
148 Red Hat "Shadowman" logos are registered trademarks of Red Hat,
149 Inc. "Linux" is a registered trademark of Linus Torvalds. All other
150 trademarks are the property of their respective owners.
Caesor@localhost Documents]$
```

Figure 10.14: Corresponding Line Number

Step 3 - Type `clear` and press **ENTER**.

Step 4 - Type file `file1.txt` and press **ENTER**. The file type is displayed, as shown in Figure 10.15.

```

Applications Places System Sun Apr 17, 3:39 AM Marcus Caesor
Caesor@localhost:~/Documents
e Edit View Search Terminal Help
Caesor@localhost Documents]$ file file1.txt
file1.txt: UTF-8 Unicode (with BOM) English text
Caesor@localhost Documents]$
```

Figure 10.15: File Type

Exercise - 4**Creating an empty file**

Step 1 - Log on to the RHEL system with the login credentials.

Step 2 - To open **Nautilus**, click **Applications → System Tools → File Browser**. Figure 10.16 displays **Nautilus**, the file manager for GNOME.

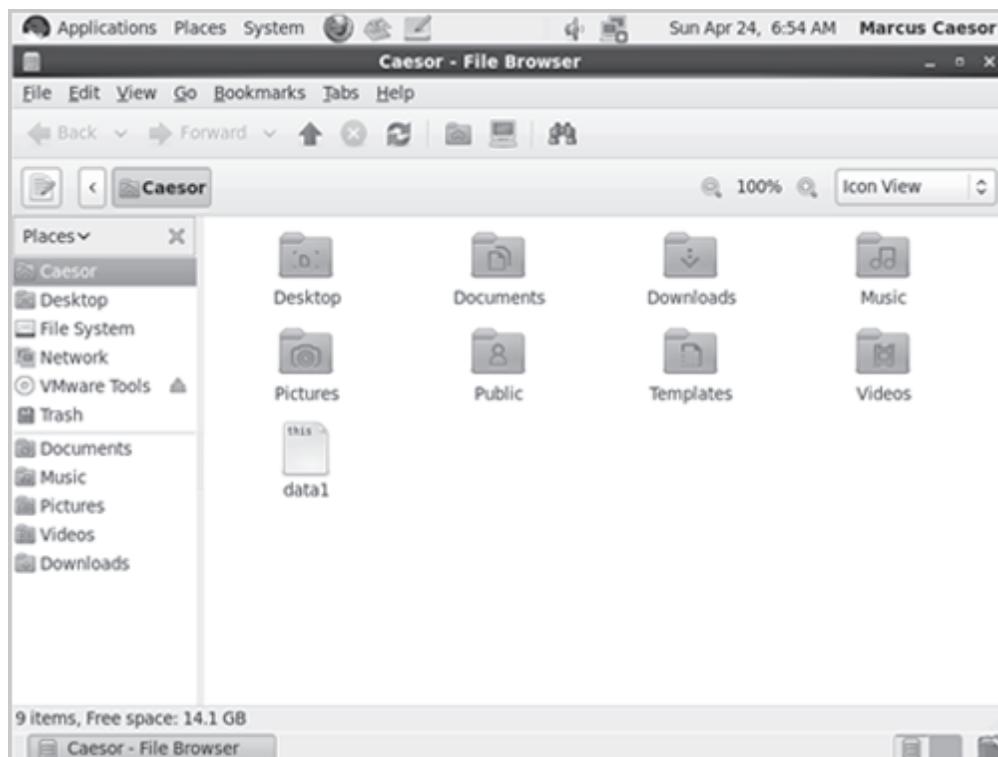


Figure 10.16: Nautilus

The system opens the default directory that is, the user's home directory.

Step 3 - To create a blank file, click **File → Create Documents → Empty File**. A new empty file named new file is created as shown in Figure 10.17.

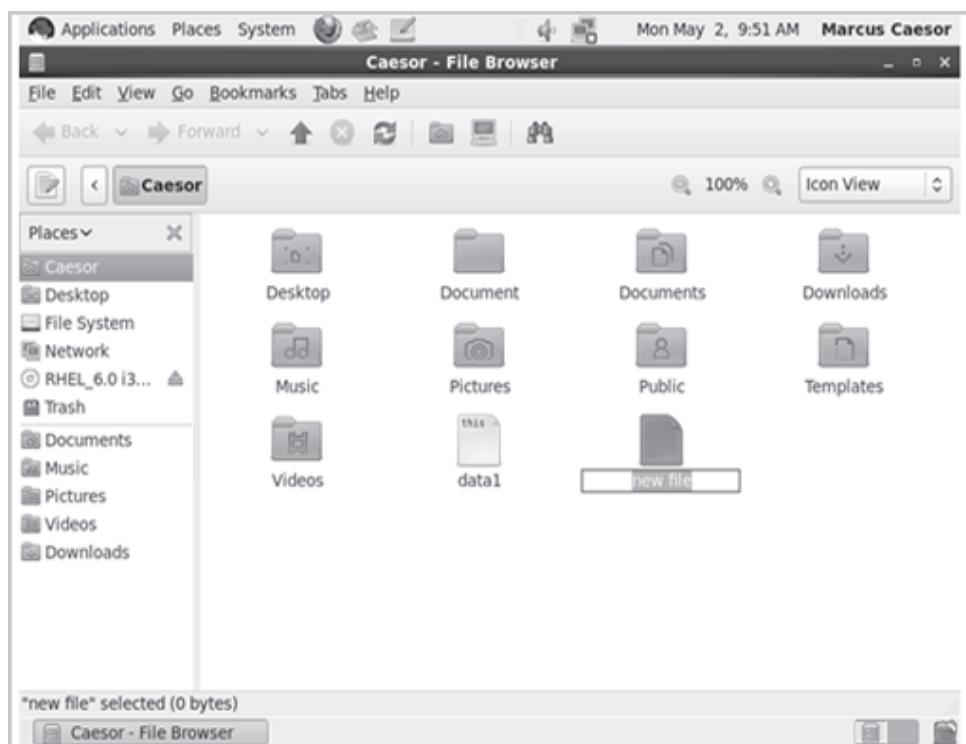


Figure 10.17: New Empty File

Step 4 - The file is renamed as **data2** as shown in Figure 10.18.

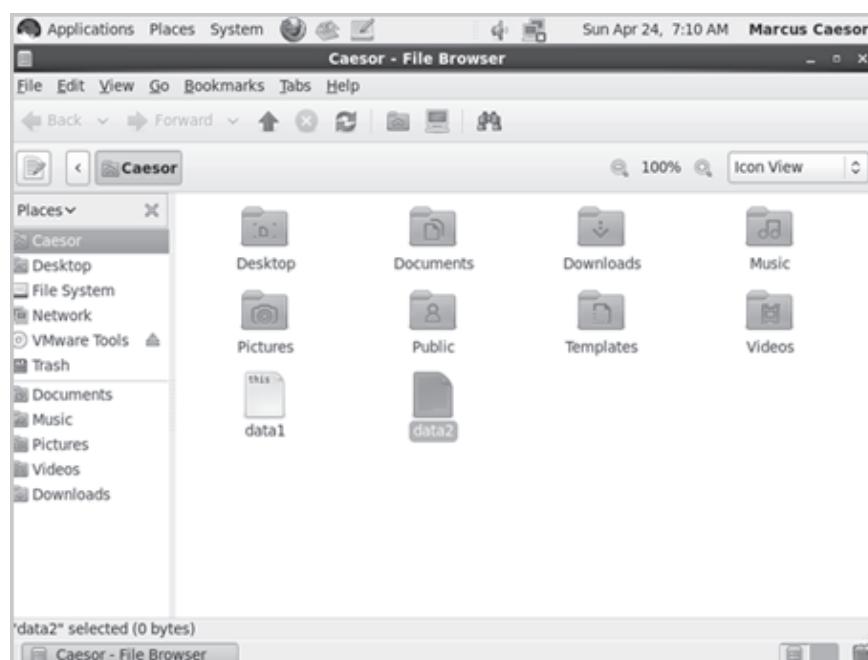


Figure 10.18: New File - data2

Users can also change the display to view the details of the files. The default display of the files and folders is set to the Icon View. Users can change the display settings to List View or Compact View.

Exercise - 5

Changing views

Step 1 - To change the view settings to List View, click the Icon View drop-down list arrow, as shown in Figure 10.19.

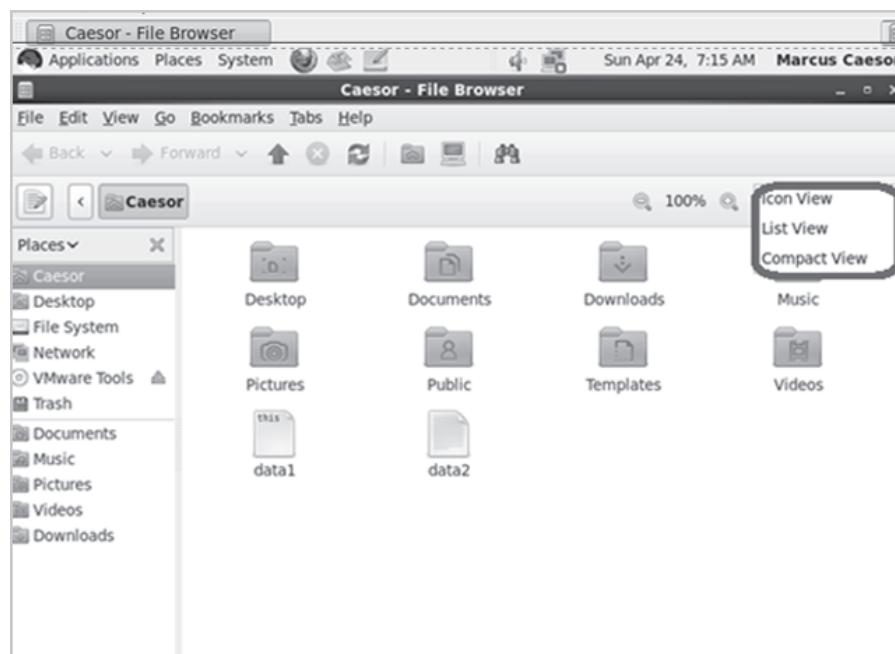


Figure 10.19: Icon View Drop-Down List

Step 2 - Click List View. The view of the files and folders changes to the list format as shown in Figure 10.20.

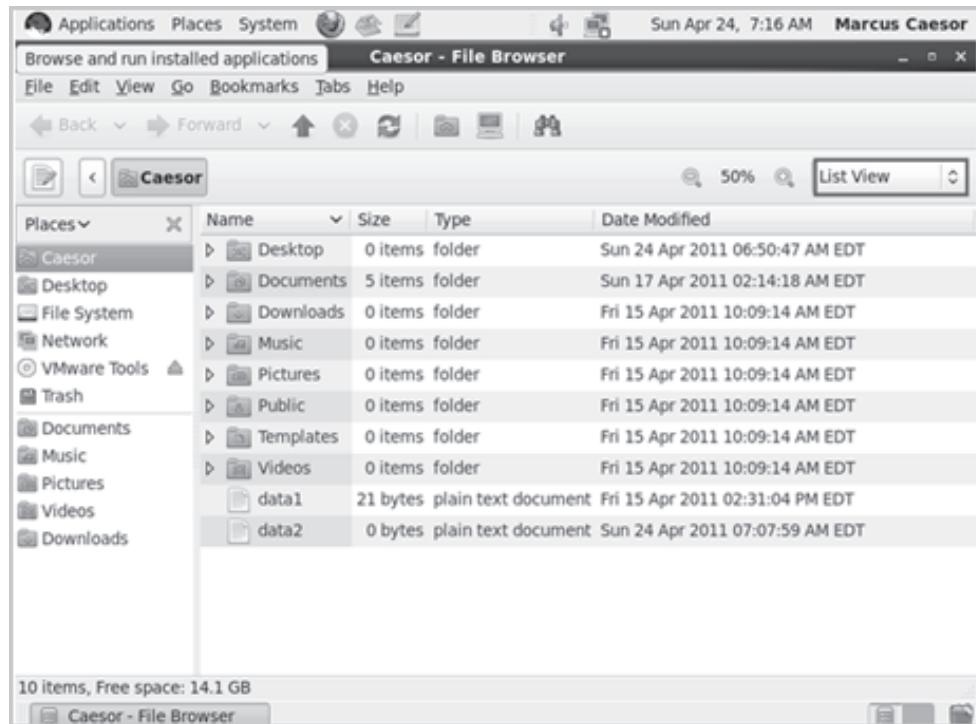


Figure 10.20: Selecting List View Format

Users can also create bookmarks to quickly browse through a few folders. There are a few default views that are listed on the bottom section of the Places side bar (Refer to Figure 10.20), which is located on the left of the file manager.

Exercise - 6

Adding a bookmark

Step 1 - To add a bookmark, on the **Bookmarks** menu, click **Add Bookmark**. The selected location, Caesor, appears in the bookmark list in the left pane as shown in Figure 10.21.

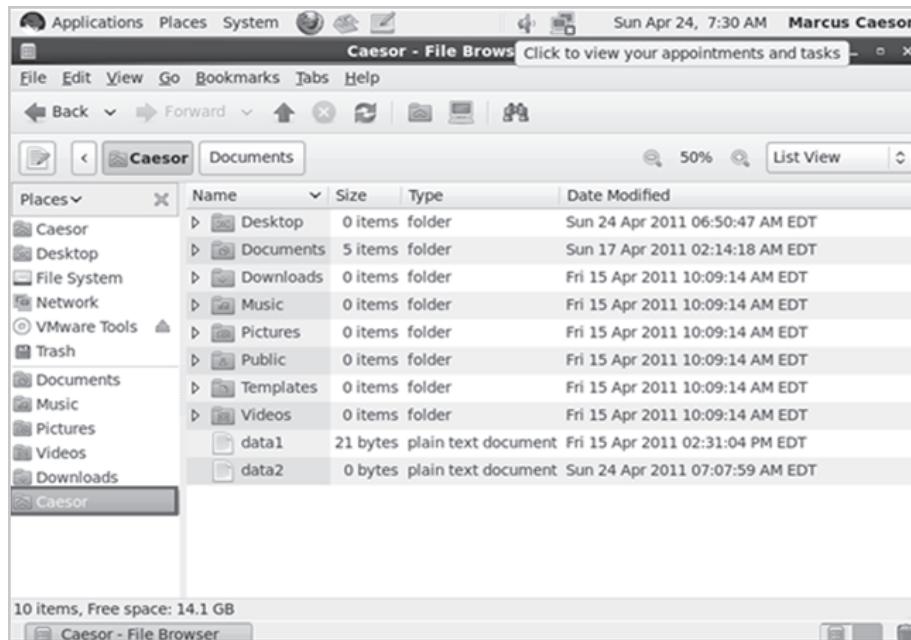


Figure 10.21: Adding Bookmark

Note - Users can also move or copy files from one location to another using the file manager.

Exercise - 7**Copying files from one directory to another directory**

Step 1 - In the left pane, click **Documents**. There are five files in the Documents folder as shown in Figure 10.22. The names of these five files are file1.txt, file2.txt, file3.txt, file4.txt and file5.txt.

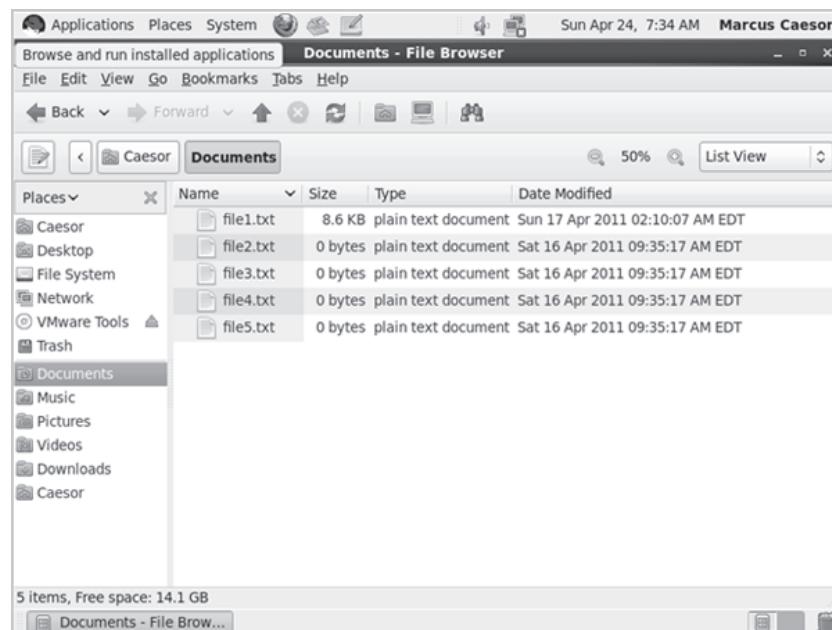


Figure 10.22: Documents Folder

Step 2 - Select all the files as shown in Figure 10.23.

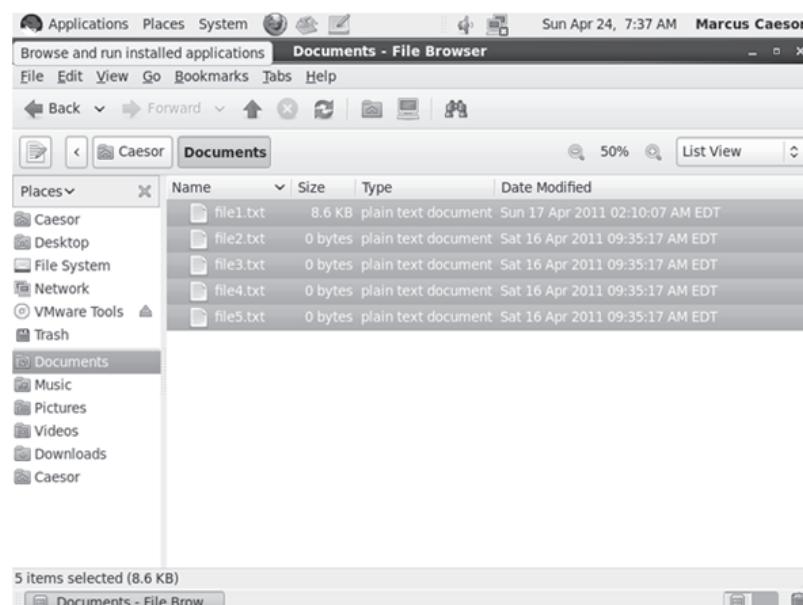


Figure 10.23: Selecting Files

Step 3 - On the **Edit** menu, click **Copy**. Alternatively, press the **CTRL+C** keys to copy all selected files.

Step 4 - In the left pane, click **Caesor**. The system displays the contents of the **Caesor** folder.

Step 5 - On the **Edit** menu, click **Paste**. Alternatively, press the **CTRL+V** keys to paste all the selected files. Figure 10.24 shows all the copied files in the **Caesor** home directory.

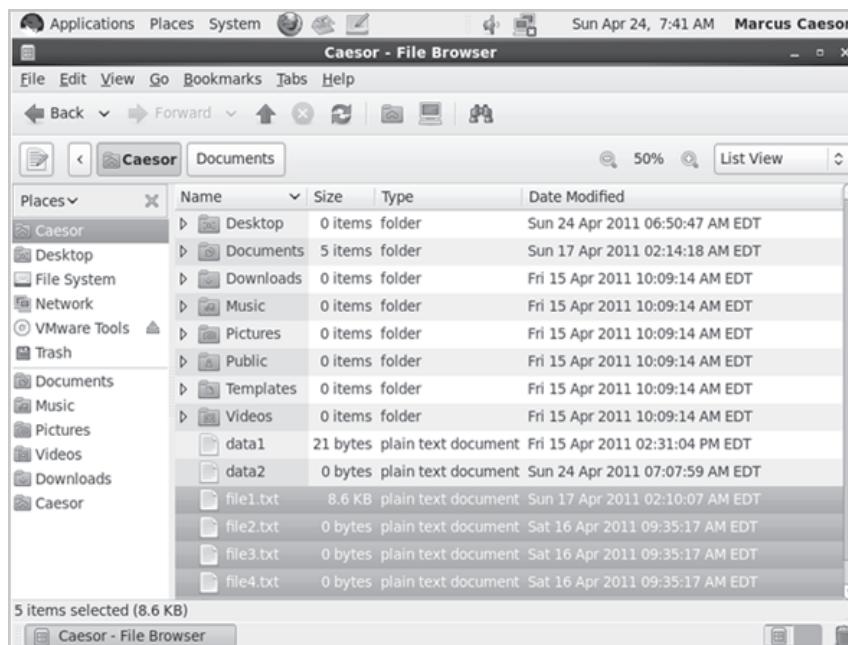


Figure 10.24: Copied Files

Note - Similarly, users can choose to move files as well.

Exercise - 8**Opening a file with gedit**

Step 1 - In the Caesor's home directory, select file1.txt.

Step 2 - On the File menu, click **Open with gedit** as shown in Figure 10.25.

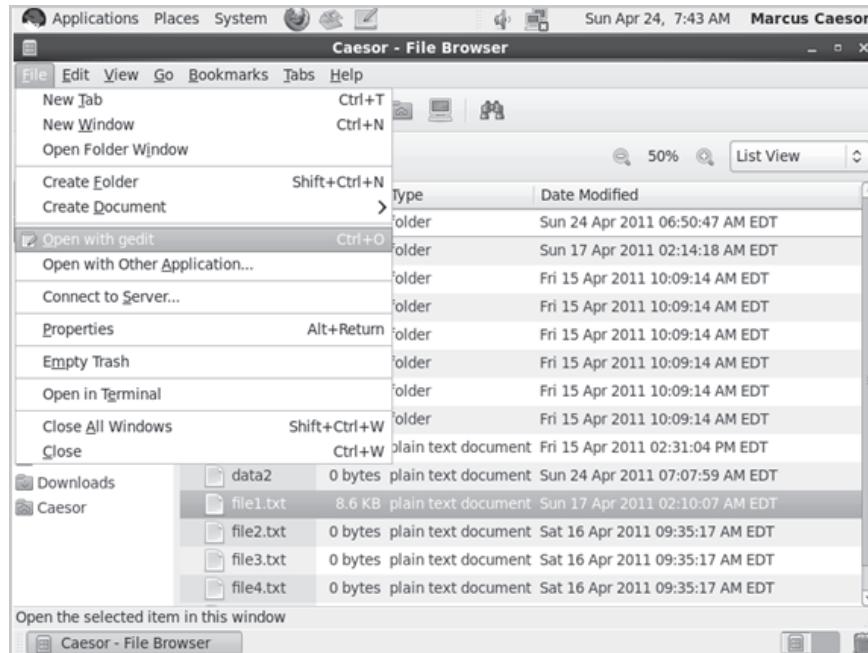


Figure 10.25: Open with gedit

Step 3 - The file opens in **gedit** as shown in Figure 10.26.

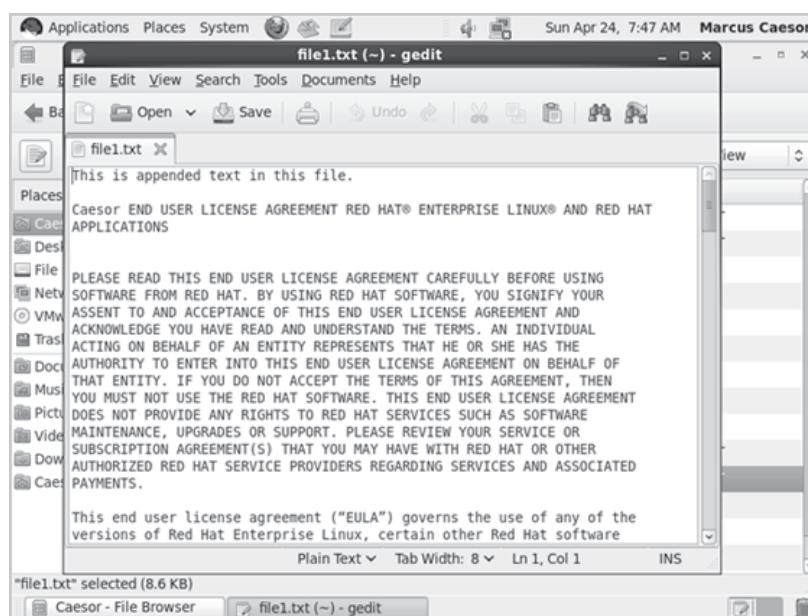


Figure 10.26: File in gedit

Step 4 - Close the opened file.

Step 5 - To close the File Browser window, click **File → Close**.

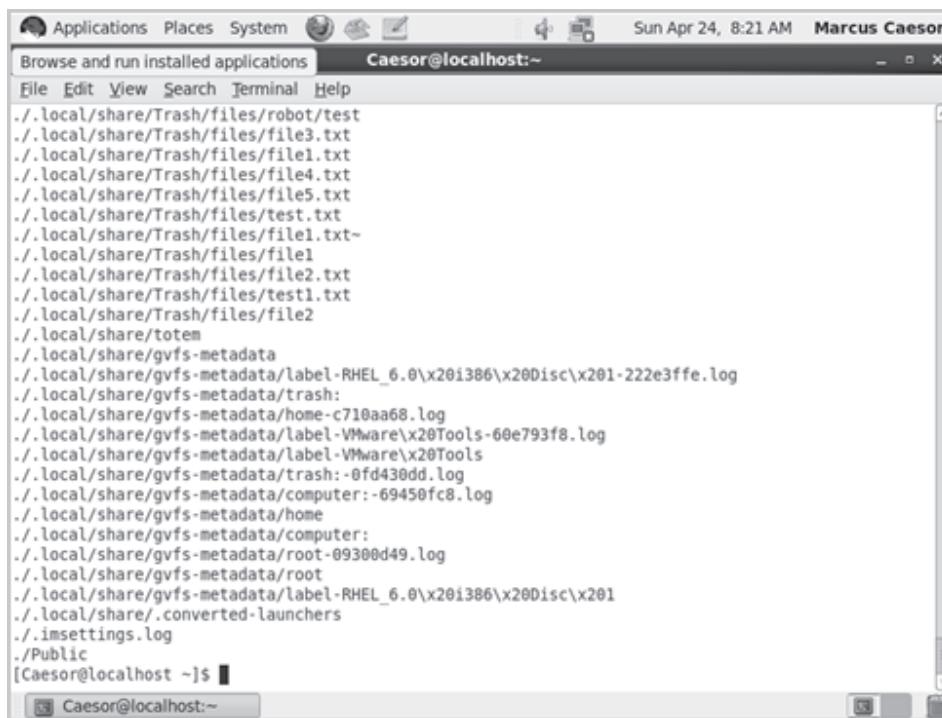
Note - With **Nautilus**, users can also open the terminal to a specific folder location, rename files and folders, connect to a server using methods such as, **ftp** or **ssh** and **compress** files and folders.

Exercise - 9

Using the **find** command to find and process files

Step 1 - Open the terminal from **Applications → System Tools → Terminal**.

Step 2 - In the terminal, type **find** and press **ENTER**. The **find** command locates all files under the current directory, which is the default directory for finding files. Figure 10.27 displays the result of **find** command.



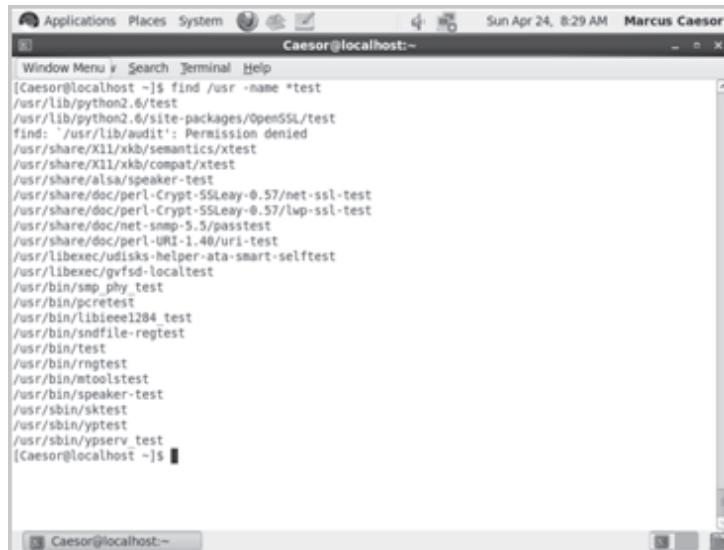
```
./local/share/Trash/files/robot/test
./local/share/Trash/files/file3.txt
./local/share/Trash/files/file1.txt
./local/share/Trash/files/file4.txt
./local/share/Trash/files/file5.txt
./local/share/Trash/files/test.txt
./local/share/Trash/files/file1.txt-
./local/share/Trash/files/file1
./local/share/Trash/files/file2.txt
./local/share/Trash/files/test1.txt
./local/share/Trash/files/file2
./local/share/totem
./local/share/gvfs-metadata
./local/share/gvfs-metadata/label-RHEL_6.0\x20i386\x20Disc\x201-222e3ffe.log
./local/share/gvfs-metadata/trash:
./local/share/gvfs-metadata/home-c710aa68.log
./local/share/gvfs-metadata/label-VMware\x20Tools-60e793f8.log
./local/share/gvfs-metadata/label-VMware\x20Tools
./local/share/gvfs-metadata/trash:-8fd430dd.log
./local/share/gvfs-metadata/computer:-69450fc8.log
./local/share/gvfs-metadata/home
./local/share/gvfs-metadata/computer:
./local/share/gvfs-metadata/root-09300d49.log
./local/share/gvfs-metadata/root
./local/share/gvfs-metadata/label-RHEL_6.0\x20i386\x20Disc\x201
./local/share/.converted-launchers
./imsettings.log
./Public
[Caesor@localhost ~]$
```

Figure 10.27: **find** Command

Step 3 - To find all files in the **/usr** directory that end with **test**, execute the following command:

```
find /usr -name *test
```

Figure 10.28 displays the result of the `find /usr -name *test` command.



```
[Caesor@localhost ~]$ find /usr -name *test
/usr/lib/python2.6/test
/usr/lib/python2.6/site-packages/OpenSSL/test
find: '/usr/lib/audit': Permission denied
/usr/share/X11/xkb/semantics/xtest
/usr/share/X11/xkb/compat/xtest
/usr/share/alsa/speaker-test
/usr/share/doc/perl-Crypt-SSTLeay-0.57/net-ssl-test
/usr/share/doc/perl-Crypt-SSTLeay-0.57/lwp-ssl-test
/usr/share/doc/net-snmp-5.5/passtest
/usr/share/doc/perl-URI-1.40/uri-test
/usr/libexec/udisks-helper-ata-smart-selftest
/usr/libexec/gvfsd-localtest
/usr/bin/smp_phy_test
/usr/bin/pcretest
/usr/bin/libieee1284_test
/usr/bin/sndfile-regtest
/usr/bin/test
/usr/bin/rngtest
/usr/bin/mtooltest
/usr/bin/speaker-test
/usr/sbin/sktest
/usr/sbin/yptest
/usr/sbin/ypserv_test
[Caesor@localhost ~]$
```

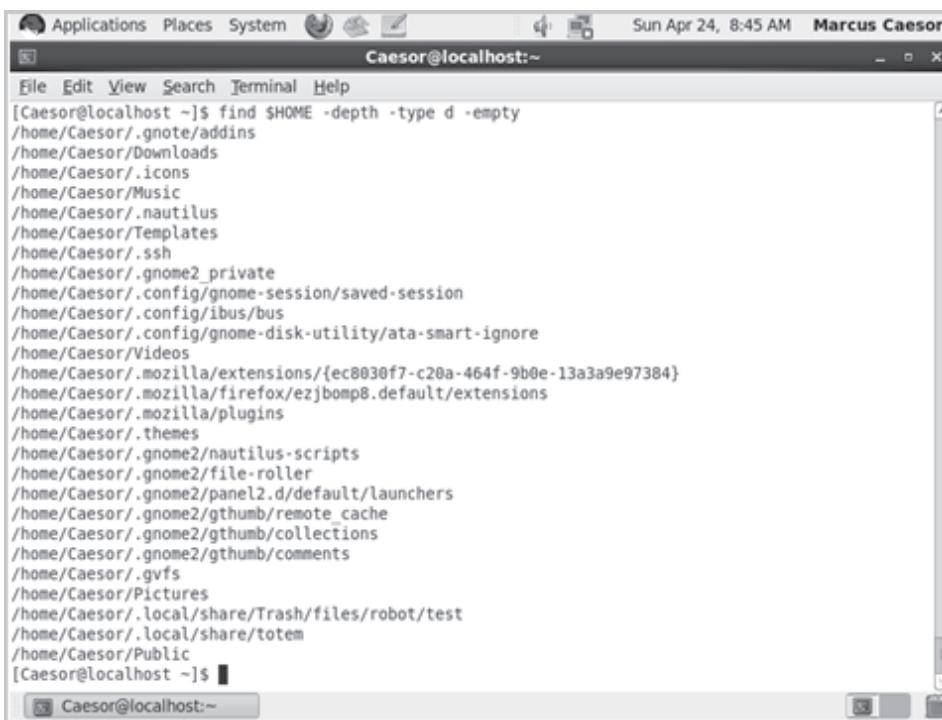
Figure 10.28: Result of the `find /usr -name *test` Command

Step 4 - Type `clear` and press **ENTER** to clear the screen.

Step 5 - To locate all the empty directories in the home directory, execute the following command:

`find $HOME -depth -type d -empty`

Figure 10.29 displays the result of the `find $HOME -depth -type d -empty` command.



```
[Caesor@localhost ~]$ find $HOME -depth -type d -empty
/home/Caesor/.gnome2
/home/Caesor/.gnote/addins
/home/Caesor/Downloads
/home/Caesor/.icons
/home/Caesor/Music
/home/Caesor/.nautilus
/home/Caesor/Templates
/home/Caesor/.ssh
/home/Caesor/.gnome2_private
/home/Caesor/.config/gnome-session/saved-session
/home/Caesor/.config/ibus/bus
/home/Caesor/.config/gnome-disk-utility/ata-smart-ignore
/home/Caesor/Videos
/home/Caesor/.mozilla/extensions/{ec8030f7-c20a-464f-9b0e-13a3a9e97384}
/home/Caesor/.mozilla/firefox/ezjbomp8.default/extensions
/home/Caesor/.mozilla/plugins
/home/Caesor/.themes
/home/Caesor/.gnome2/nautilus-scripts
/home/Caesor/.gnome2/file-roller
/home/Caesor/.gnome2/panel2.d/default/launchers
/home/Caesor/.gnome2/gthumb/remote_cache
/home/Caesor/.gnome2/gthumb/collections
/home/Caesor/.gnome2/gthumb/comments
/home/Caesor/.gvfs
/home/Caesor/Pictures
/home/Caesor/.local/share/Trash/files/robot/test
/home/Caesor/.local/share/totem
/home/Caesor/Public
[Caesor@localhost ~]$
```

Figure 10.29: Locating Empty Directories in Home Directory

Step 6 - To find the file 'file1.txt' in the home directory and its subdirectories, execute the following command:

```
find $HOME -name file1.txt
```

Figure 10.30 display the result of running the find \$HOME -name file1.txt command.

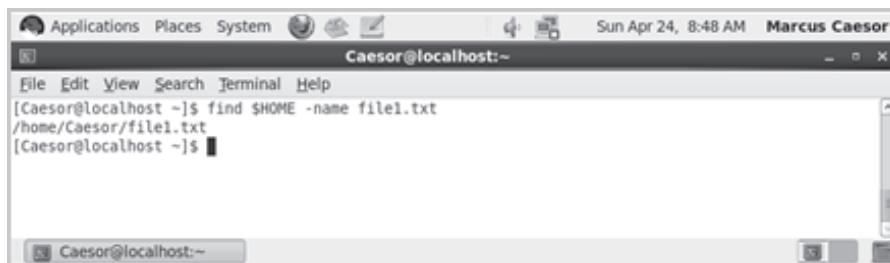
A screenshot of a terminal window titled "Caesor@localhost:~". The window shows the command "find \$HOME -name file1.txt" being run, and it outputs two results: "/home/Caesor/file1.txt" and "/home/Caesor/Downloads/file1.txt". The terminal window has a standard Linux-style interface with a title bar, menu bar, and scroll bars.

Figure 10.30: Locating a Specific File (file1.text) in Subdirectories

Step 7 - To locate the file 'file1.txt' in the home directory, execute the following command:

```
find $HOME -user Caesor -name file1.txt
```

Figure 10.31 display the result of the find \$HOME -user Caesor -name file1.txt command.

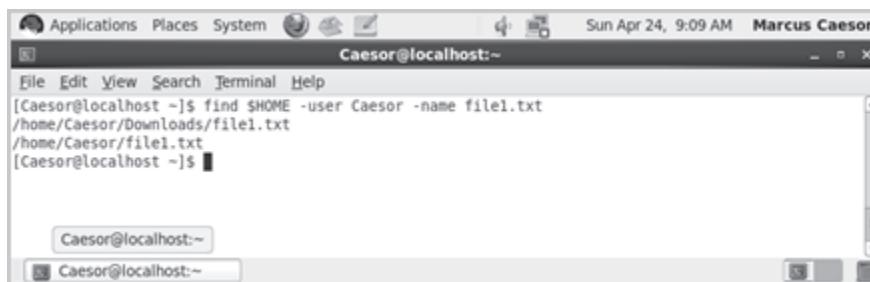
A screenshot of a terminal window titled "Caesor@localhost:~". The window shows the command "find \$HOME -user Caesor -name file1.txt" being run, and it outputs two results: "/home/Caesor/Downloads/file1.txt" and "/home/Caesor/file1.txt". The terminal window has a standard Linux-style interface with a title bar, menu bar, and scroll bars.

Figure 10.31: Using find Command to Locate a File

Step 8 - Type clear and press **ENTER** to clear the screen.

Step 9 - Type exit and press **ENTER** to close the terminal.

Exercise - 10**Checking current swap space**

Step 1 - To invoke the command line terminal, click **Applications → System Tools → Terminal**.

Step 2 - Execute the following command to check the swap space as shown in Figure 10.32. `cat /proc/swaps`

```
[Caesor@localhost ~]$ cat /proc/swaps
Filename      Type      Size    Used   Priority
/dev/dm-1     partition 2064376 0       -1
[Caesor@localhost ~]$
```

Figure 10.32: Checking Swap Space

An alternative command, `swapon`, displays the same result as shown in Figure 10.33.

```
[Caesor@localhost ~]$ swapon -s
Filename      Type      Size    Used   Priority
/dev/dm-1     partition 2064376 0       -1
[Caesor@localhost ~]$
```

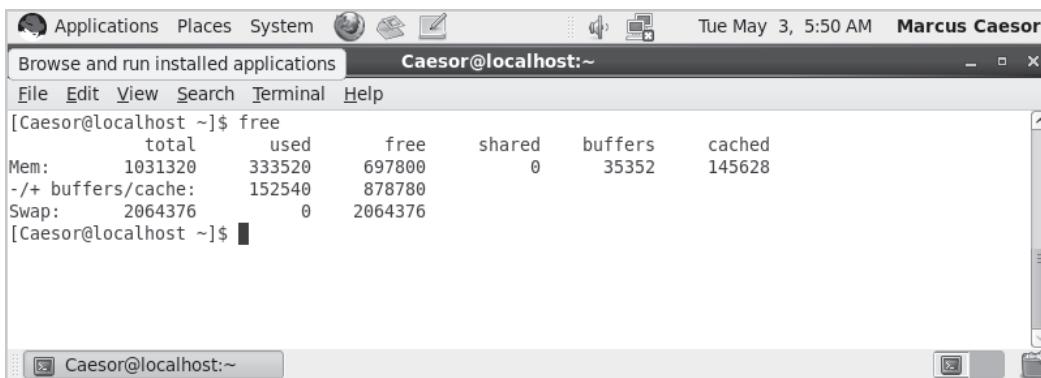
Figure 10.33: Using `swapon` Command to Check Swap Space

Another alternative command that provides a result on memory and swap space is the `free` command.

Execute the following command:

```
free
```

Figure 10.34 displays the memory and swap space using `free` command.



A screenshot of a terminal window titled "Caesor@localhost:~". The window shows the output of the "free" command. The output is as follows:

```
[Caesor@localhost ~]$ free
total        used        free      shared      buffers      cached
Mem:   1031320    333520    697800          0     35352    145628
-/+ buffers/cache:  152540    878780
Swap:  2064376       0    2064376
[Caesor@localhost ~]$
```

Figure 10.34: Memory and Swap Space

Exercise - 11**Creating a swap space**

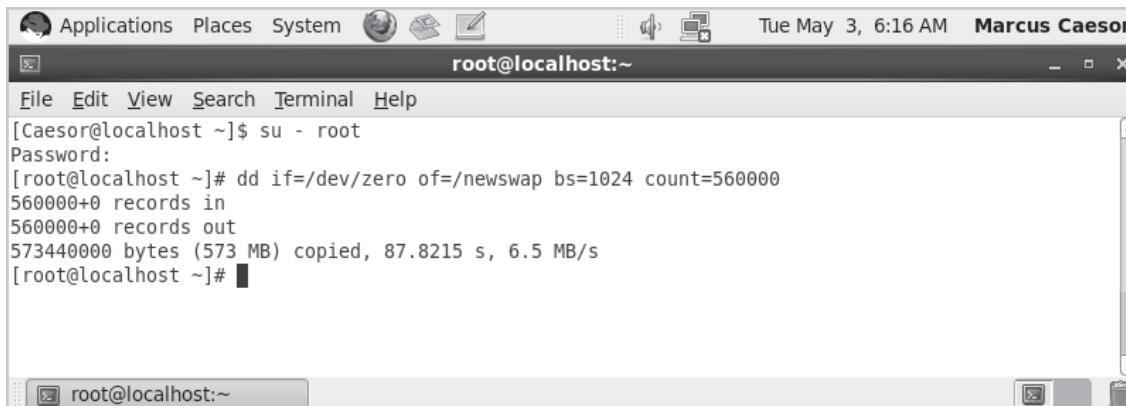
Users can create additional swap space to the system by creating a file and assigning it as swap. Begin by creating the swap file using the dd command. The size can be changed by adjusting the count=variable; the following creates a 131 MB file.

Note - Users require superuser or root user privileges to create a swap space.

Step 1 - At the terminal, execute the `su` command to log in as a root user. Provide the password for the root user.

Step 2 - To create a swap file of 563 MB, execute the following command (Refer to Figure 10.35):

```
dd if=/dev/zero of=/newswap bs=1024 count=560000
```



A screenshot of a terminal window titled "root@localhost:~". The window shows the root user executing the "dd" command to create a swap file. The command is:

```
[Caesor@localhost ~]$ su - root
Password:
[root@localhost ~]# dd if=/dev/zero of=/newswap bs=1024 count=560000
560000+0 records in
560000+0 records out
573440000 bytes (573 MB) copied, 87.8215 s, 6.5 MB/s
[root@localhost ~]#
```

Figure 10.35: Creating the Swap File

Step 3 - After creating the swap file, a user must configure it as swap space. Execute the following command:

```
mkswap /newswap
```

Figure 10.36 displays the result of this command.

```
[Caesor@localhost ~]$ su - root
Password:
[root@localhost ~]# dd if=/dev/zero of=/newswap bs=1024 count=560000
560000+0 records in
560000+0 records out
573440000 bytes (573 MB) copied, 87.8215 s, 6.5 MB/s
[root@localhost ~]# mkswap /newswap
mkswap: /newswap: warning: don't erase bootbits sectors
on whole disk. Use -f to force.
Setting up swap space version 1, size = 559996 KiB
no label, UUID=f12ee825-4439-41b6-afa5-568fc4651424
[root@localhost ~]#
```

Figure 10.36: Configuring Swap Space

Step 4 - Add the swap space to the system in real-time. For this, execute the following command:

```
swapon /newswap
```

The new swap space is now added to the system in real-time.

Step 5 - Create an entry in the /etc/fstab file. With this entry in the fstab file, the new swap space is automatically detected when the system boots.

```
/newswap swap swap defaults 0 0
```

The swap space is now created and configured.

Step 6 - Close the terminal with the exit command.

Exercise - 12

Creating ext3 partition

Step 1 - Log on to the RHEL system with the login credentials.

Step 2 - Click Applications → System Tools → Terminal to open the terminal.

Step 3 - Execute the su command to log in as root user. Provide the password for the root user.

Step 4 - Execute the following command to verify whether the new disk exists in the system:

```
fdisk -l
```

After executing this command, the user is prompted with a message that the new disk does not contain a valid partition table as shown in Figure 10.37.

```

Applications Places System Wed Apr 27, 8:05 AM Marcus Caesor
Browse and run installed applications root@localhost:~
File Edit View Search Terminal Help
[root@localhost ~]# fdisk -l

Disk /dev/sdb: 4294 MB, 4294967296 bytes
255 heads, 63 sectors/track, 522 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disk /dev/sdb doesn't contain a valid partition table

Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000cb5a9

Device Boot Start End Blocks Id System
/dev/sdal * 1 64 512000 83 Linux
Partition 1 does not end on cylinder boundary.
/dev/sda2 64 2611 20458496 8e Linux LVM

Disk /dev/dm-0: 18.8 GB, 18832424960 bytes
255 heads, 63 sectors/track, 2289 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

```

Figure 10.37: Prompted Message

Step 5 - At the command prompt, execute the following command:

fdisk /dev/sdb

Figure 10.38 displays the output of **fdisk /dev/sdb** command.

```

Applications Places System Wed Apr 27, 8:09 AM Marcus Caesor
Browse and run installed applications root@localhost:~
File Edit View Search Terminal Help
[root@localhost ~]# fdisk /dev/sdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0xa4854c76.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition 4 will be corrected by w(rite)

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
switch off the mode (command 'c') and change display units to
sectors (command 'u').

Command (m for help): 

```

Figure 10.38: Output of **fdisk /dev/sdb** Command

Step 6 - Type **n** at the command prompt and press **ENTER**.

A prompt is displayed. The user must either create a primary or extended partition as shown in Figure 10.39.

```
[root@localhost ~]# fdisk /dev/sdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0xa4854c76.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
switch off the mode (command 'c') and change display units to
sectors (command 'u').

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
```

Figure 10.39: Creating a Primary or Extended Partition

Step 7 - Type **p** and press **ENTER**. The user is prompted to choose a value from 1 to 4. This defines the first cylinder as shown in Figure 10.40.

```
[root@localhost ~]# fdisk /dev/sdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0xa4854c76.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

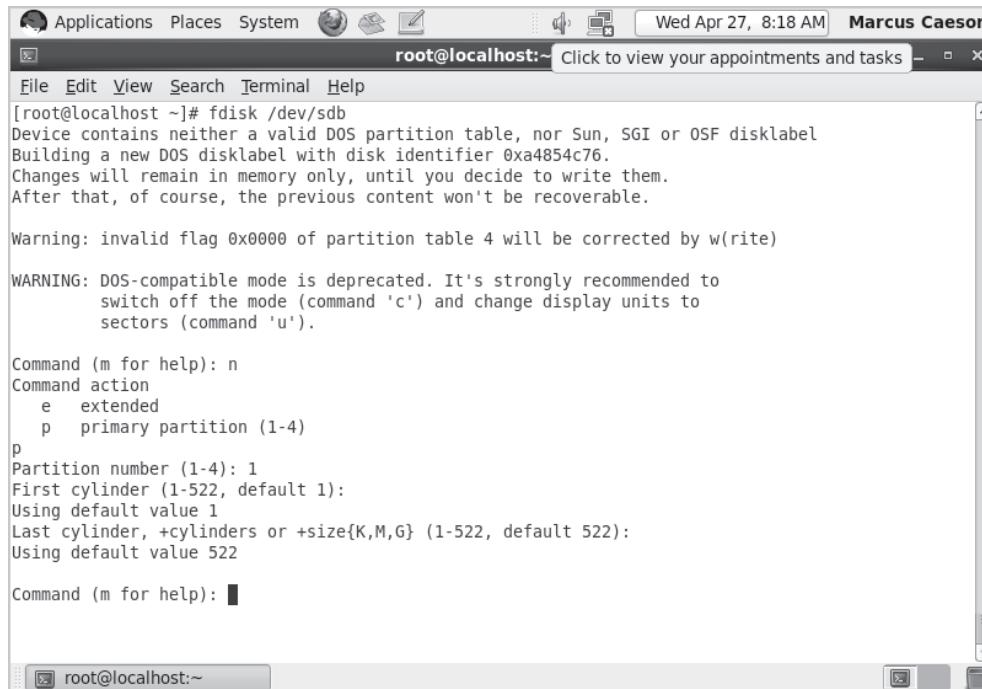
WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
switch off the mode (command 'c') and change display units to
sectors (command 'u').

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-522, default 1):
```

Figure 10.40: Defining First Cylinder

Step 8 - Press **ENTER** to accept default value.

Step 9 - Press **ENTER** to confirm and accept the default value to create first and last cylinder as shown in Figure 10.41.



```
[root@localhost ~]# fdisk /dev/sdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0xa4854c76.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

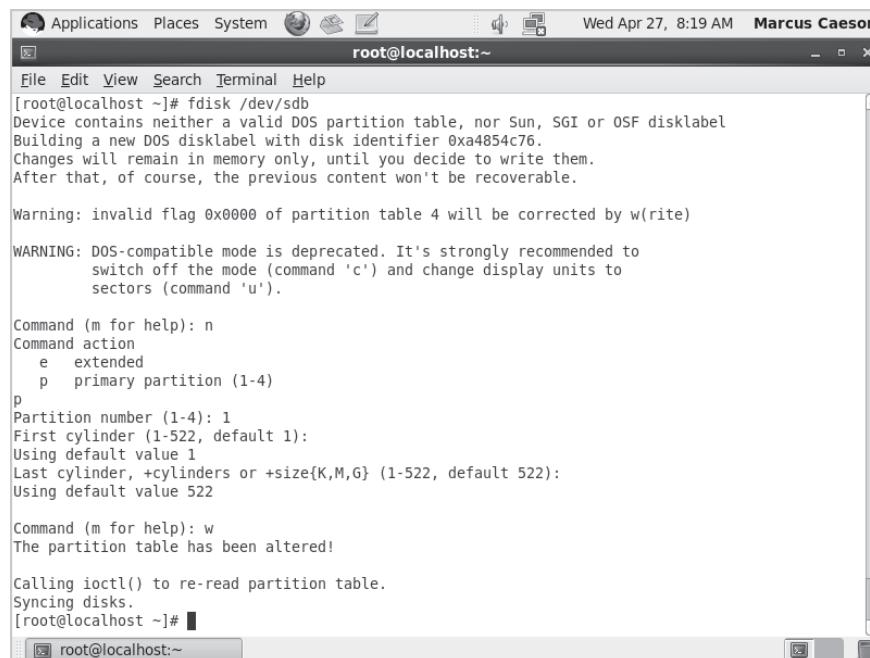
WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
         switch off the mode (command 'c') and change display units to
         sectors (command 'u').

Command (m for help): n
Command action
      e   extended
      p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-522, default 1):
Using default value 1
Last cylinder, +cyinders or +size{K,M,G} (1-522, default 522):
Using default value 522

Command (m for help):
```

Figure 10.41: Accepting Default Values to Create First and Last Cylinder

Step 10 - To write the changes to the disk, type **w**. Changes are saved to the disk as shown in Figure 10.42.



```
[root@localhost ~]# fdisk /dev/sdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0xa4854c76.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
         switch off the mode (command 'c') and change display units to
         sectors (command 'u').

Command (m for help): n
Command action
      e   extended
      p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-522, default 1):
Using default value 1
Last cylinder, +cyinders or +size{K,M,G} (1-522, default 522):
Using default value 522

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
[root@localhost ~]#
```

Figure 10.42: Saving Changes

Step 11 - Restart computer for changes to take effect.

Note - Alternate to restarting the system is to execute the *partprobe* command.

Step 12 - Log on to the RHEL system with the login credentials.

Step 13 - Click Applications → System Tools → Terminal to open terminal.

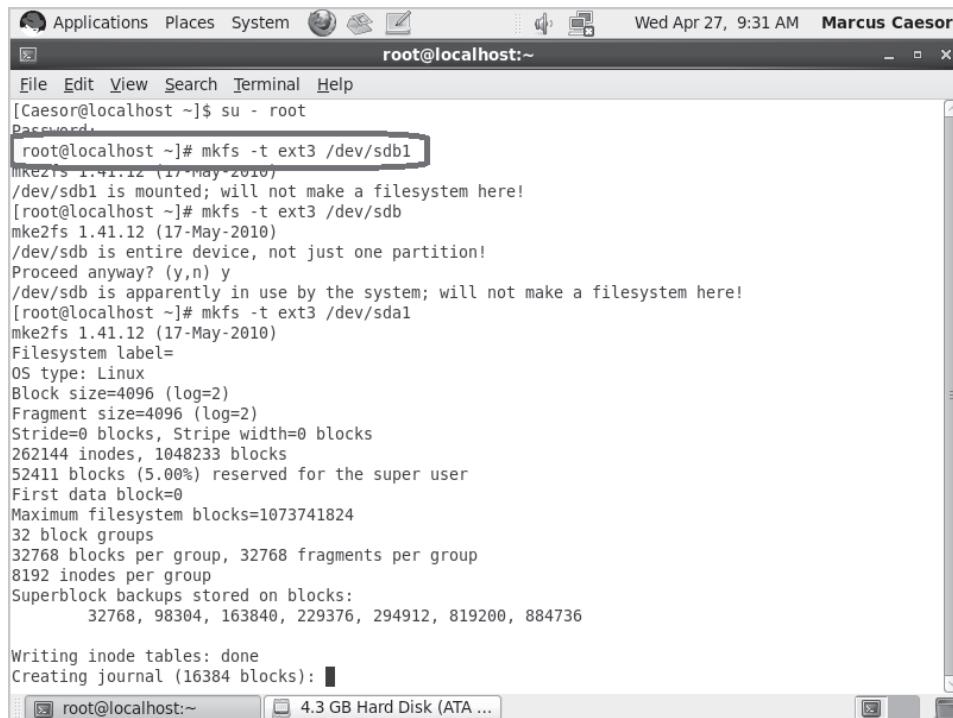
Step 14 - Execute the *su* command to log in root user. Provide the password for the root user.

Step 15 - Execute the following command:

```
mkfs -t ext3 /dev/sda
```

Note - The device name can differ in different systems.

Figure 10.43 displays the output of the *mkfs -t ext3 /dev/sda* command.



The screenshot shows a terminal window titled "root@localhost:~". The window contains the following text output from the *mkfs -t ext3 /dev/sda* command:

```
[Caesar@localhost ~]$ su - root
Password:
[root@localhost ~]# mkfs -t ext3 /dev/sdb1
mke2fs 1.41.12 (17-May-2010)
/dev/sdb1 is mounted; will not make a filesystem here!
[root@localhost ~]# mkfs -t ext3 /dev/sdb
mke2fs 1.41.12 (17-May-2010)
/dev/sdb is entire device, not just one partition!
Proceed anyway? (y,n) y
/dev/sdb is apparently in use by the system; will not make a filesystem here!
[root@localhost ~]# mkfs -t ext3 /dev/sda1
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
262144 inodes, 1048233 blocks
52411 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1073741824
32 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912, 819200, 884736

Writing inode tables: done
Creating journal (16384 blocks): [■]
[root@localhost ~]# 4.3 GB Hard Disk (ATA ...]
```

Figure 10.43: Output of the *mkfs -t ext3 /dev/sda* Command

Step 16 - To make the mounting of this partition permanent open */etc/fstab* file use the following command:

```
vi /etc/fstab
```

Step 17 - To make a permanent mount, type /dev/sdb1 /home/mydisk ext3 defaults 1 2 and press **ENTER** as shown in Figure 10.44.

Figure 10.44: Executing /dev/sdb1 /home/mydisk ext3 defaults 1 2 Command

After completing the steps to create a new partition, the user must format the partition to the ext3 file system.



To add on to your knowledge of the subject,
visit the **REFERENCES** page



11.1 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.

11.1.1 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.

11.1.2 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:

- 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.

11.2 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`, `csh`, `ksh`, `sh` or `zsh`. For example, the `csh` 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.

11.2.1 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:

→ **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 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:

- * - 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 11.1.

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 11.1: 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 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:

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 split into shell words.
7. File globs (*, ?, [abc]) are expanded.
8. The command is executed.

11.3 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:

- #!/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.

11.3.1 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 11.2.

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

Table 11.2: 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

11.3.2 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 11.3.

Option	Description
-a fname	The words are assigned to sequential indices of the array variable fname, starting at 0

Option	Description
-d delim	The character specified in delim is used to terminate the input line, rather than newline character
-e	If the standard input is coming from a terminal, readline is used to obtain the
-n nchars	Read returns after reading n number of characters
-s	Silent mode. If input is coming from a terminal, characters are not echoed

Table 11.3: 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"
```

11.4 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"
```

11.4.1 **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) 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
```

11.5 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

11.6 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.1st`. 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.1st
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
```

11.6.1 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
```

11.6.2 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 along with 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
```

11.6.3 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
```

11.6.4 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
```

11.7 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 begins 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.

```
$ anymore "Want to continue"  
Want to continue ?(y/n) : n  
$ echo $1  
1
```

It displays the value returned in the default pattern.

11.8 Check Your Progress

1. The command to display the formatted output is _____.

(A)	output	(C)	printf
(B)	echo	(D)	print

2. The _____ command forces the loop to jump to its next iteration.

(A)	loop	(C)	break
(B)	continue	(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	(C)	if/else
(B)	while-loop	(D)	case

4. Which of the following are selection structures?

(A)	for-loop	(C)	if/else
(B)	while-loop	(D)	case

5. Which of the following provides the concept of code reusability?

(A)	function	(C)	repetition
(B)	recursive	(D)	command

11.8.1 Answers

1.	C
2.	B
3.	D
4.	D
5.	A



Summary

- 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.

TechnoWise



Are you a
TECHIE GEEK
looking for updates?

Logon to

www.onlinevarsity.com

Session - 12

Shell Scripting (LAB)

Welcome to the Session, **Shell Scripting (LAB)**.

In this session, users will learn to create shell scripts, use various structure language statements, repetitive structures, and functions.

In this Session, you will learn to:

- Create a basic shell script
- Use the case, if/else, and while structures in shell script
- Use functions in shell script



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 12.1.**

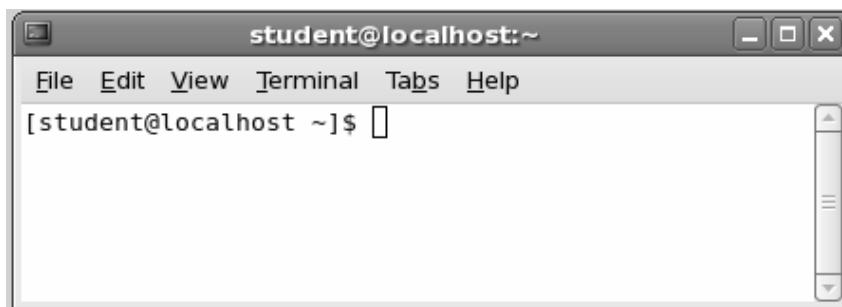


Figure 12.1: Terminal Window

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 12.2.**

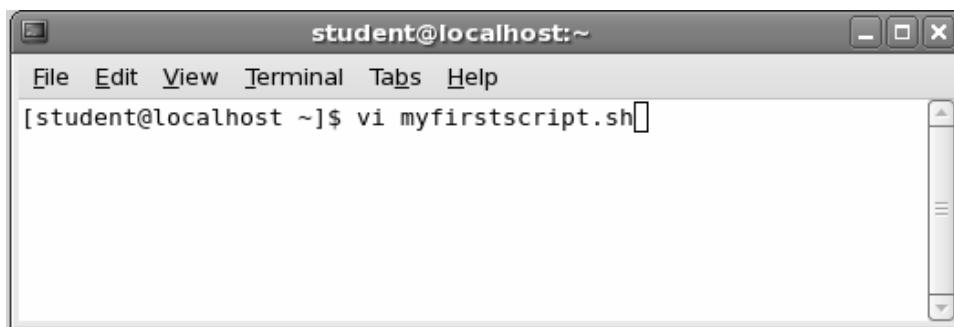
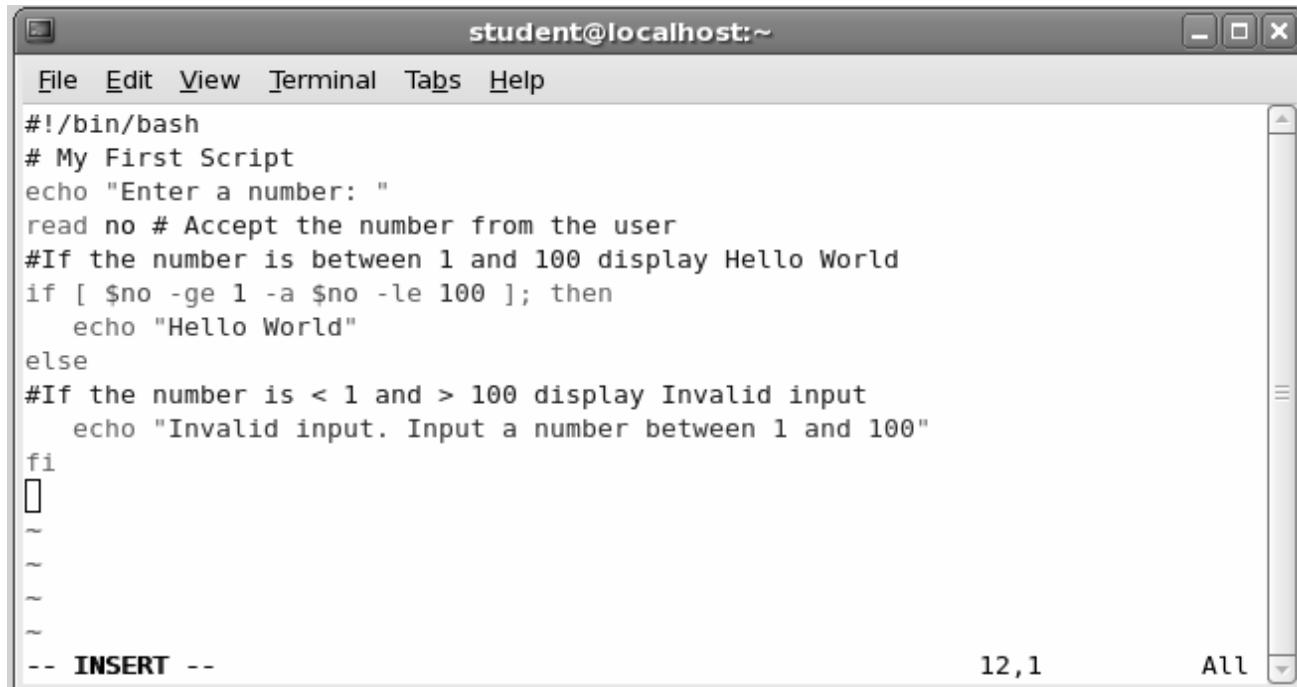


Figure 12.2: Creating a New Shell Script

2. Press **i** to insert text at the current cursor position.
3. Type the following script as shown in Figure 12.3.



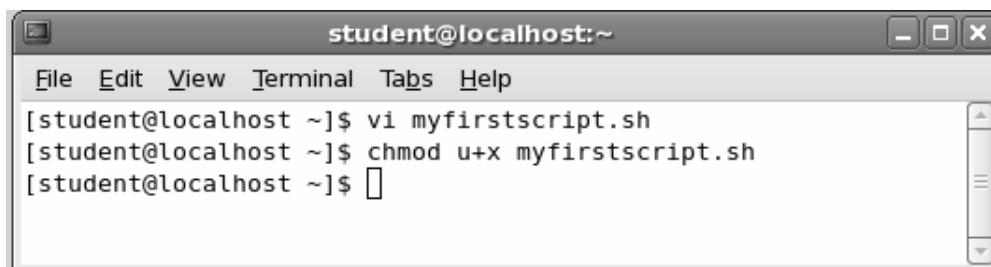
```

student@localhost:~$ vi myfirstscript.sh
File Edit View Terminal Tabs Help
#!/bin/bash
# My First Script
echo "Enter a number: "
read no # Accept the number from the user
#If the number is between 1 and 100 display Hello World
if [ $no -ge 1 -a $no -le 100 ]; then
    echo "Hello World"
else
#If the number is < 1 and > 100 display Invalid input
    echo "Invalid input. Input a number between 1 and 100"
fi
~
~
~
~
-- INSERT --
12,1 All

```

Figure 12.3: `myfirstscript.sh`

4. Type **:wq**, to save and quit the file.
5. Make the script executable using the `chmod` command as shown in Figure 12.4.



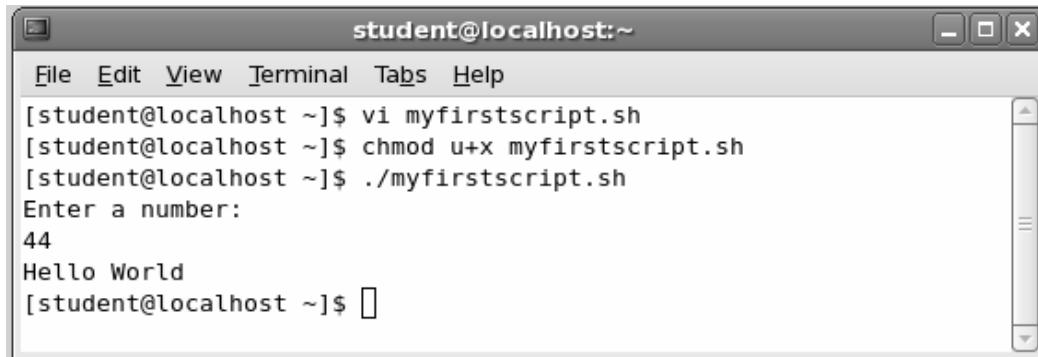
```

student@localhost:~$ vi myfirstscript.sh
[student@localhost ~]$ chmod u+x myfirstscript.sh
[student@localhost ~]$

```

Figure 12.4: 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 12.5.

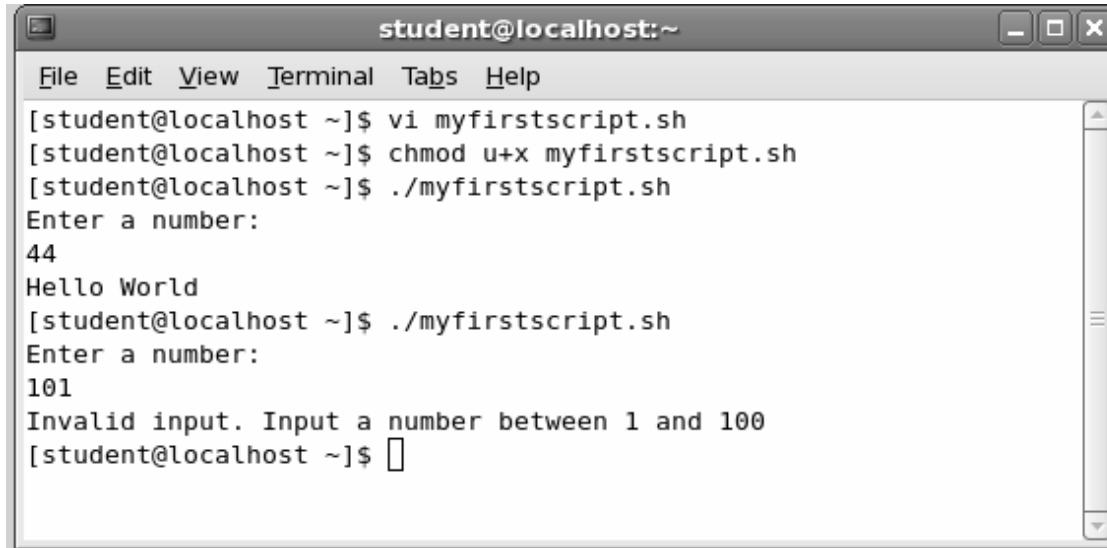
```
student@localhost:~$ vi myfirstscript.sh
student@localhost:~$ chmod u+x myfirstscript.sh
student@localhost:~$ ./myfirstscript.sh
Enter a number:
44
Hello World
student@localhost:~$
```

Figure 12.5: 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 12.6.



```
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 12.6: 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 example, 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 12.7.

```
student@localhost:~$ vi reverse.sh
#!/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"

```

Figure 12.7: reverse.sh

2. Execute the script.

The output is as shown in Figure 12.8.

```
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 12.8: 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.

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:

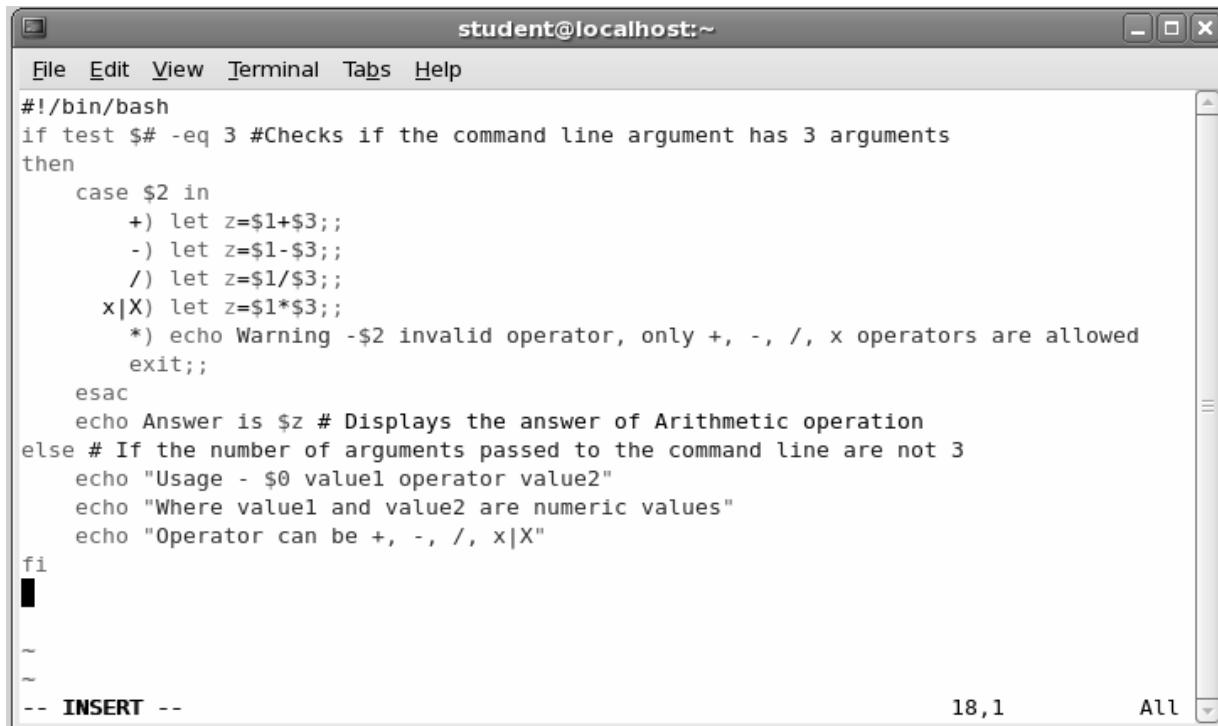
The second argument passed to the command line is the operator. Match the operator passed against the cases in the case structure and perform the appropriate arithmetic operations. The pattern for each of them use the appropriate arithmetic symbols namely, +, -, /, * respectively.

If any other operator is entered, display an error message.

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

- Create the file calculator.sh in vi editor and type the statements as shown in Figure 12.9.**



```

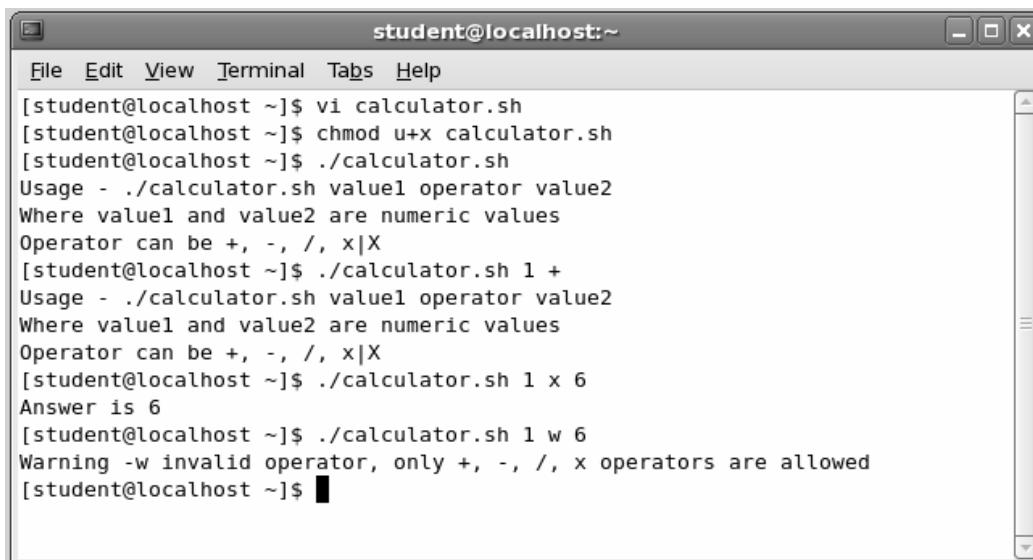
student@localhost:~$ vi calculator.sh
File Edit View Terminal Tabs Help
#!/bin/bash
if test $# -eq 3 #Checks if the command line argument has 3 arguments
then
    case $2 in
        +) let z=$1+$3;;
        -) let z=$1-$3;;
        /) let z=$1/$3;;
        x|X) let z=$1*$3;;
        *) echo Warning -$2 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 --

```

Figure 12.9: calculator.sh

- Execute the script.**

The output of the various operations is as shown in Figure 12.10.



```

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 12.10: Output of calculator.sh

Exercise 4:**Using Functions****Problem**

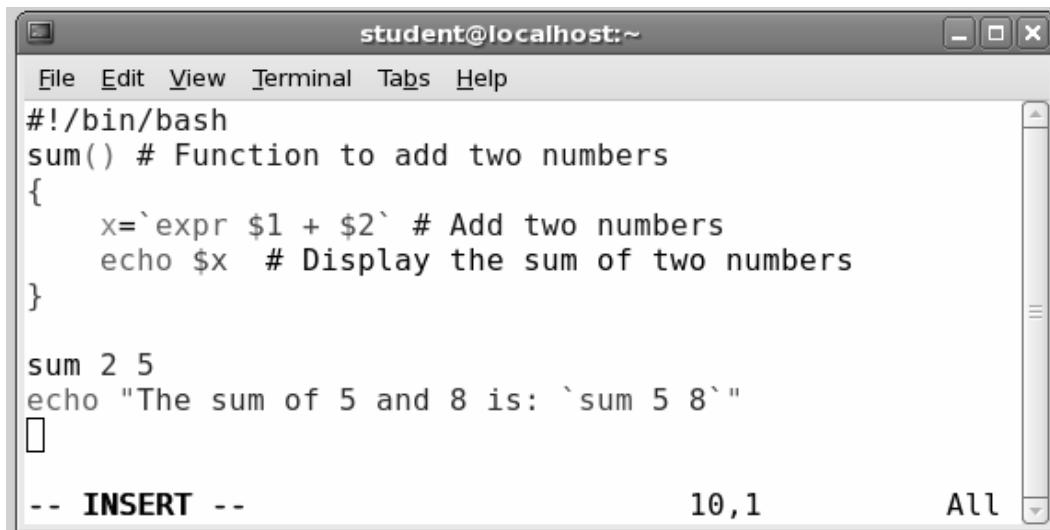
Write a script to add two numbers using functions.

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 12.11.



```
student@localhost:~
```

```
#!/bin/bash
sum() # Function to add two numbers
{
    x=`expr $1 + $2` # Add two numbers
    echo $x # Display the sum of two numbers
}

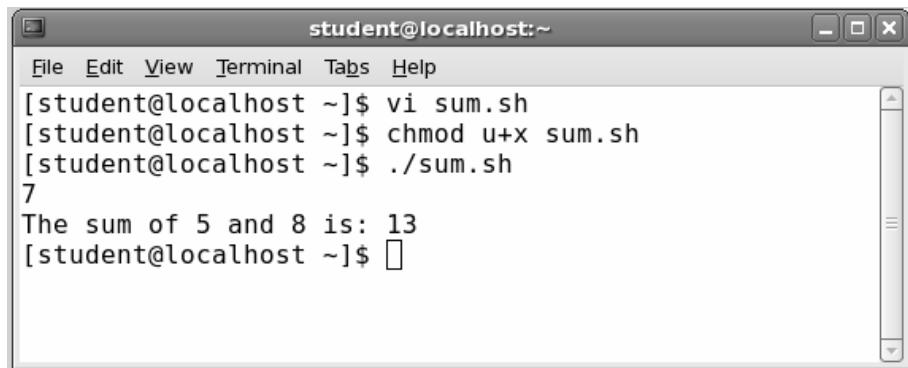
sum 2 5
echo "The sum of 5 and 8 is: `sum 5 8`"
-- INSERT --          10,1      All
```

Figure 12.11: 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 12.12.



A screenshot of a terminal window titled "student@localhost:~". The window has a standard title bar with icons for minimize, maximize, and close. The menu bar includes "File", "Edit", "View", "Terminal", "Tabs", and "Help". The main area of the terminal shows the following command-line session:

```
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 12.12: Output of sum.sh

GROWTH
RESEARCH
OBSEvation
UPDATES
PARTICIPATION



13.1 Network

If there are two or more computers connected together, it is considered to be a network. There are a number of tasks that a user performs when connected to the network. Users can browse the Internet or use the remote printers that are connected and configured on a print server.

13.1.1 Printing

RHEL uses Common UNIX Printing System (CUPS) as the default printing system. CUPS use the Internet Printing Protocol (IPP) for broadcasting the shared printers and allow other users to use this information to connect to the shared printers. The other users on the network can browse through the shared printers. The remote computers that are connecting to the shared printers do not require printer configuration. The shared printers that other computers connect to are shown under the Remote Printers section in Printer Configuration.

The following are the three ways in which printers can be managed:

- By editing the `printers.conf` configuration file in the `/etc/cups` directory. This file can be modified to allow only a few printers to the public.
- By using the Printer Configuration tools that can be invoked from the **System → Administration → Printing**.
- By invoking the `system-config-printer` tool from the command prompt.

RHEL offers printers to be added to the computers in two different ways that are as follows:

- **Individual printer** - A single printer is added to a computer. If the printer goes offline or is unavailable, the computer is unable to print.
- **Printer class** - A group of printers are added to the computer as a single printer. The class can have both local and remote printers. Instead of a single printer, the class is defined as the default printer. The printer that can print fast is used for printing. If a printer is busy, it is not used for printing when a print command is given.

Note - The log files for the CUPS printing system are located in the `/var/log/cups/` directory.

13.1.2 Web Browser

The default Web browser for the RHEL operating system is Mozilla Firefox. The version that is pre-installed with RHEL is 3.6.9. Figure 13.1 shows Mozilla Firefox window.



Figure 13.1: Mozilla Firefox

Mozilla Firefox has a number of features that help users to browse the Web easily. Some of the features are as follows:

- Bookmarking
- History
- Add-ons
- Error Console
- Popup blocker
- Private browsing

RHEL also provides the capability of browsing the Web from the command prompt. A user installs elinks package to browse using the command line browser.

After the elinks package is installed, execute the following command to browse at the command prompt:

```
#links <Website name>  
#links http://www.redhat.com
```

The command line Web browser also supports frames and Secure Socket Layer (SSL).

13.2 System Monitor

RHEL allows users to manage and monitor system using the System Monitor, which can be accessed from the **Applications → System Tools** menu. A user can use the System Monitor to keep a watch over the processes that are running, resources that are being utilized and file system status, such as free space.

13.2.1 Monitoring System Resources

The CPU, memory, network and disk are considered to be the key system resources. It is essential for any system administrator to manage and monitor them on regular intervals. If the monitoring is not done, servers are most likely to face problem. For example, a system administrator has to continuously monitor the server resources that are being utilized. If in a situation, where the system resource such as RAM is being over utilized, the system administrator can use the System Monitor to observe the memory usage and take necessary actions.

In the System Monitor, the Resources tab (Refer to Figure 13.2) displays the following three key resources that are being monitored:

- CPU usage
- Memory and Swap file usage
- Network usage

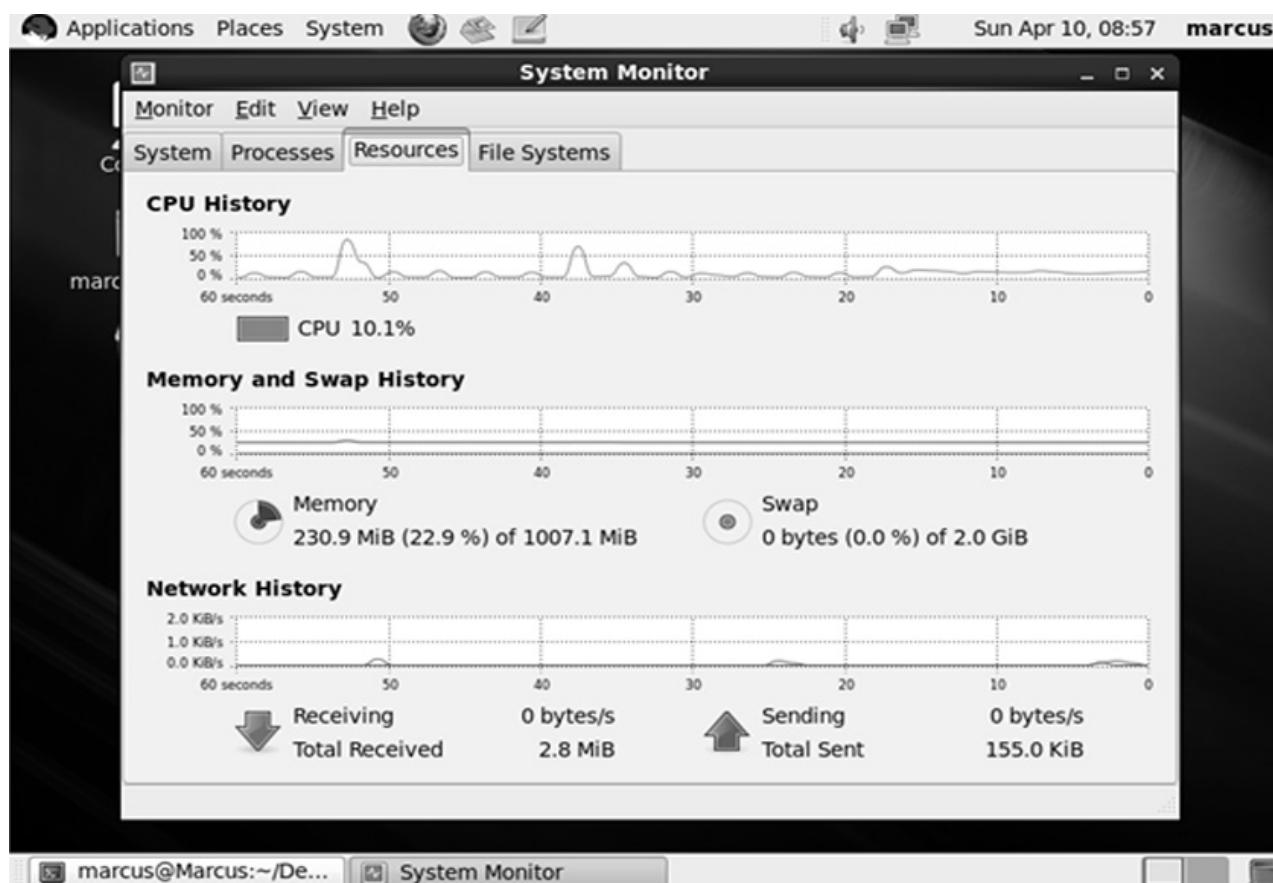


Figure 13.2: Resources Tab

The Processes tab (Refer to Figure 13.3) displays the list of processes that are running. Using this tab, a user can change the priority of a process, kill a process or stop and start a process.

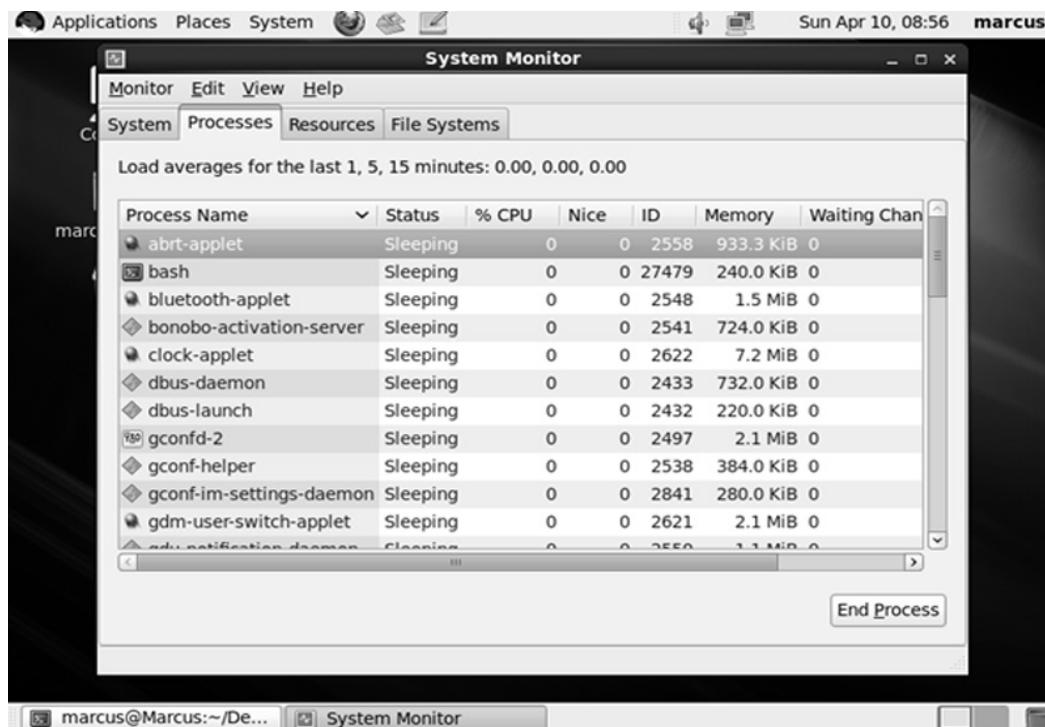


Figure 13.3: Processes Tab

The File Systems tab (Refer to Figure 13.4) displays the status of the available file systems.

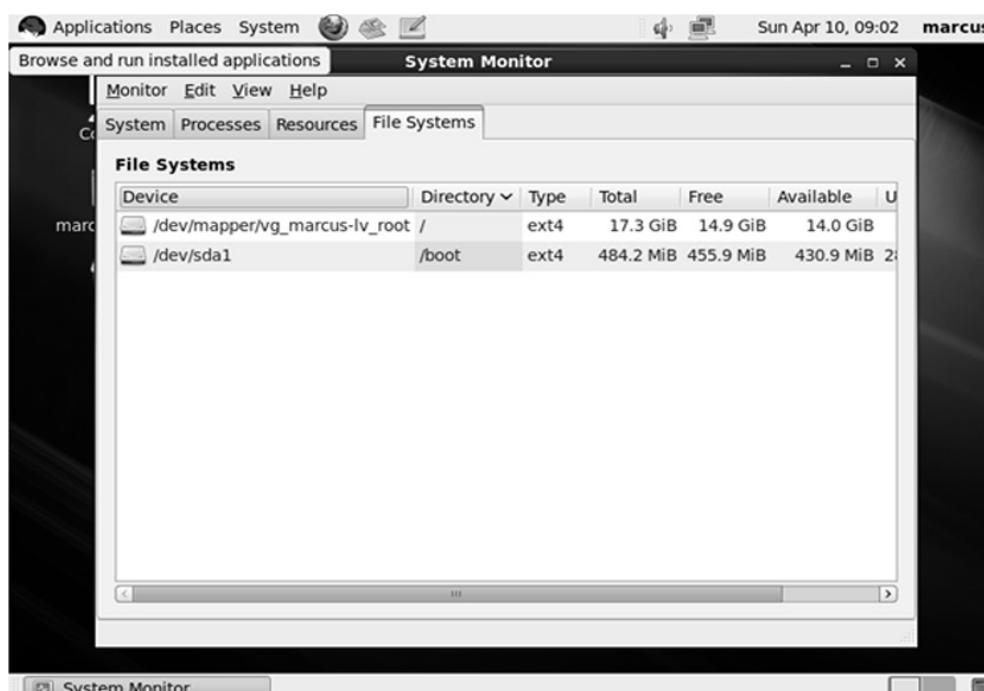


Figure 13.4: File Systems Tab

A user can also check the status of the processes from the command prompt using the `top` command. To view the current processes, type `top` at the command prompt. Figure 13.5 shows a sample output of the `top` command.

```

marcus@Marcus:~/Desktop
File Edit View Search Terminal Help
top - 09:27:08 up 4:49, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 138 total, 1 running, 137 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.3%us, 0.3%sy, 0.0%ni, 99.3%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1031320k total, 932124k used, 99196k free, 71840k buffers
Swap: 2064376k total, 0k used, 2064376k free, 628456k cached

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1188 dbus 20 0 14860 1924 996 S 0.3 0.2 0:02.44 dbus-daemon
2339 root 20 0 49988 20m 8592 S 0.3 2.1 0:47.81 Xorg
 1 root 20 0 2824 1408 1204 S 0.0 0.1 0:01.70 init
  2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
  3 root RT 0 0 0 0 S 0.0 0.0 0:00.00 migration/0
  4 root 20 0 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/0
  5 root RT 0 0 0 0 S 0.0 0.0 0:00.00 watchdog/0
  6 root 20 0 0 0 0 S 0.0 0.0 0:00.07 events/0
  7 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuset
  8 root 20 0 0 0 0 S 0.0 0.0 0:00.00 khelper
  9 root 20 0 0 0 0 S 0.0 0.0 0:00.00 netns
 10 root 20 0 0 0 0 S 0.0 0.0 0:00.00 async/mgr
 11 root 20 0 0 0 0 S 0.0 0.0 0:00.00 pm
 12 root 20 0 0 0 0 S 0.0 0.0 0:00.00 sync_supers
 13 root 20 0 0 0 0 S 0.0 0.0 0:00.00 bdi-default
 14 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kintegrityd/0
 15 root 20 0 0 0 0 S 0.0 0.0 0:00.78 kblockd/0

```

Figure 13.5: Output of `top` Command

13.2.2 Managing System Software

Software updates can be applied to the RHEL operating system either locally or through the RHN Web site. `yum` helps in uploading the software updates to the RHEL operating system. It can be configured to receive updates from software repositories that are located on the Internet or an intranet. A user must be logged on as a root user to run the `yum` command.

`yum` has a number of advantages that help in providing software updates and they are as follows:

- Provides updates according to the hardware architecture or software version. For example, if RHEL is installed on x86 server, the software updates is more relevant to the x86 architecture and not to x64.
- Fetches software updates from various locations. `yum` can automatically search through the repositories and provide the relevant update.
- Available in two different formats, graphical and command line.

Note - To use the repositories at the RHN Web site, the RHEL installation must be registered and activated with the RHN Web site.

To run the updates from the command prompt, execute the following command:

```
#yum install <pkgname>
```

A user can choose to install either specific hardware architecture or a particular version of the application.

A user can also perform software updates from the graphical interface. A user can use the Add/Remove Software utility that can be invoked from **System → Administration**.

Figure 13.6 shows the **Add/Remove Software** utility window.

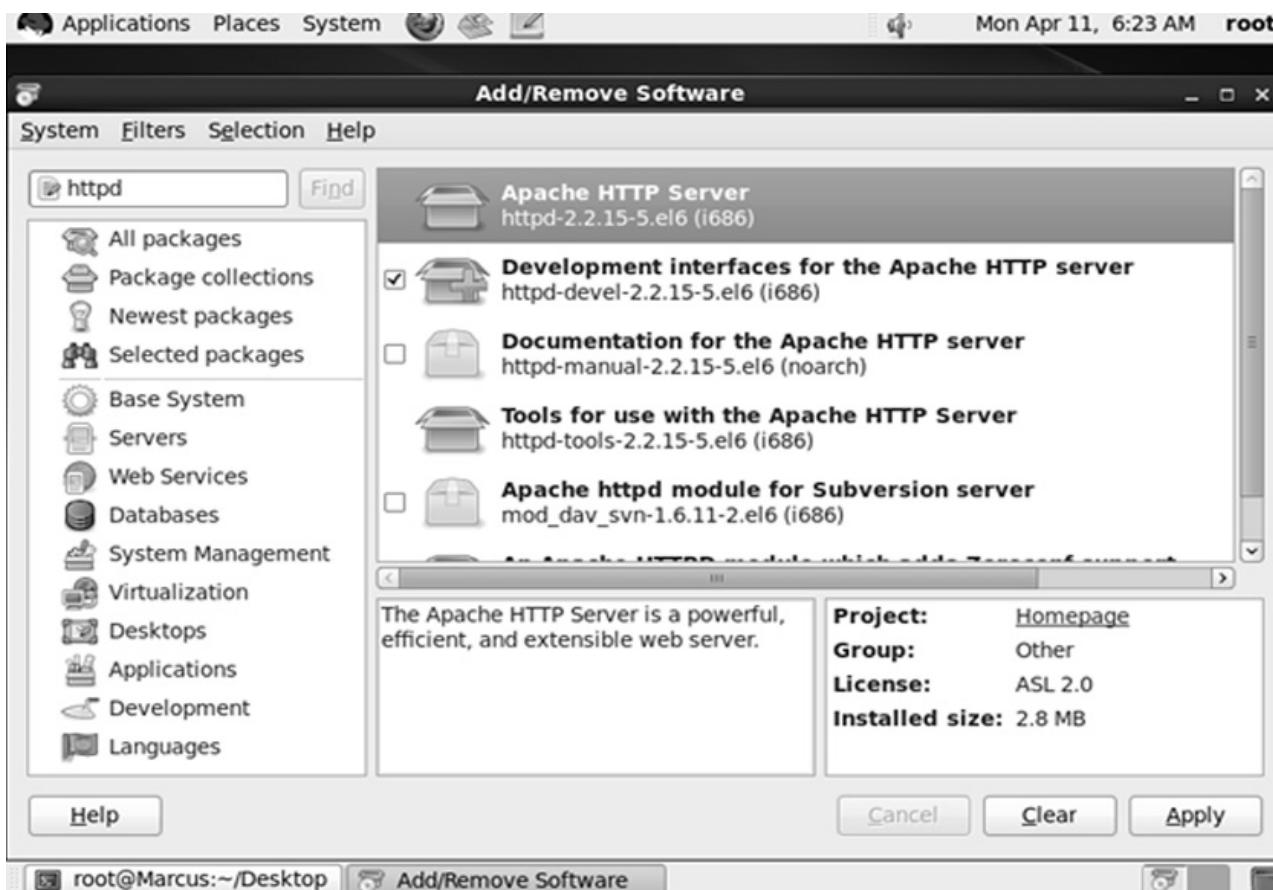


Figure 13.6: Add/Remove Software Utility Window

To install any updated software or add new software, a user must select the package and click **Apply**. If there are any additional packages that are to be installed, the user is prompted to first install those packages, and then install the main package.

13.3 Establishing Network Connectivity

A user can use three different methods to establish network connectivity. The following tasks can be performed by the users:

- Add a new network adapter
- Modify the existing network adapter properties
- Assign an IP address to the network adapter
- Assign or change DNS server

The three different methods that a user can use to configure the network are as follows:

- Configuring network at the command prompt
- Configuring network in graphical environment (X Window System or GUI tool)
- Configuring network configuration files

Note - All three methods provide the same end result. They are the preferable choices to configure the network connectivity on a RHEL system.

13.4 Administering Remote Systems

In a network environment, it is essential for an administrator to manage multiple computers. The administrator can have physical access to the computers. If the computers are not physically accessible, the administrator should remotely administer these computers. Consider an example, an organization where Mary is the system administrator. The organization has offices at 34 different locations. The servers that are running RHEL 6.0 are located at four different locations. Mary is the only system administrator who manages the servers. In this situation, Mary should either travel to each of the locations or manage these servers remotely.

RHEL provides multiple tools that users can use to remotely manage computers that can be located far from where the administrator is located. Some of the tools are as follows:

- Secure Shell
- Rsync
- Nautilus

Secure Shell

Secure Shell (SSH) is part of the OpenSSH suite of tools that are included in the RHEL operating system. Earlier, administrators used Telnet to remotely manage computers. However, the biggest problem with Telnet was transmission of data in plain text format. There was no method of encrypting data during transmission.

There are two components in OpenSSH. These two components are as follows:

- The first component is the server, which is the computer that is being connected.
- The second component is the client, which is the computer that is initiating the action to the server.

Note - A system can be both an SSH server and a client. A computer that is accepting incoming connections can also initiate a request to connect to another computer.

The **ssh** command is as follows:

```
ssh [user@]hostname
ssh [user@]hostname [command]
```

The complete command is shown in Figure 13.7.

```
[root@localhost ~]# ssh
usage: ssh [-1246AaCfgKkMNnqsTtUvXxYy] [-b bind_address] [-c cipher_spec]
           [-D [bind_address:]port] [-e escape_char] [-F configfile]
           [-i identity_file] [-L [bind_address:]port:host:hostport]
           [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port]
           [-R [bind_address:]port:host:hostport] [-S ctl_path]
           [-w local_tun[:remote_tun]] [user@]hostname [command]
[root@localhost ~]# _
```

Figure 13.7: File B ssh Command

A computer that is configured as the OpenSSH server accepts the incoming requests on port 22. If port 22 is not opened, the user must ensure that this port is enabled to accept incoming requests. To use SSH, the user must install the OpenSSH package. The user should then install the openssh-server RPM package. The user must enable the service if it is already not enabled.

To enable the service, execute the following command:

```
service sshd start
```

To check the status of the service, execute the following command as shown in Figure 13.8:

```
service sshd status
```

```
[root@localhost ~]# service sshd start
[root@localhost ~]# service sshd status
openSSH-daemon (pid 1493) is running...
[root@localhost ~]# _
```

Figure 13.8: Checking the SSH Status

After the connection has been made, `scp` can be used for secure file transfer.

→ Rsync

`Rsync` is a utility that helps in backing up data on local and remote computers. Using `rsync`, a user can copy data from local computers to the remote computers. The user can also copy data between two directories on a local computer. The key advantage with `rsync` is that it only copies the difference between the files if there is any kind of duplicity.

Consider an example of using `rsync` on the user's local computer. The user can connect to a file server to backup critical files. The user can backup files only on Friday evenings. All the files that a user changes or creates on a local computer can be backed up. `Rsync` compares all the files that already exist in the backup folder and then copies only the differences.

Figure 13.9 displays the `rsync` command with its parameters.

```
Usage: rsync [OPTION]... SRC [SRC]... DEST
      or   rsync [OPTION]... SRC [SRC]... [USER@]HOST:DEST
      or   rsync [OPTION]... SRC [SRC]... [USER@]HOST::DEST
      or   rsync [OPTION]... SRC [SRC]... rsync://[USER@]HOST[:PORT]/DEST
      or   rsync [OPTION]... [USER@]HOST:SRC [DEST]
      or   rsync [OPTION]... [USER@]HOST::SRC [DEST]
      or   rsync [OPTION]... rsync://[USER@]HOST[:PORT]/SRC [DEST]
```

Figure 13.9: `rsync` Command

The parameters for `rsync` command are as follows:

- **Options** - These are the parameters that are passed to the command. For example, the `-a` parameter is for the archive mode. It preserves the directories, subdirectories, symbolic links, permissions, time stamps and groups and file permissions including ownership.
- **First directory** - This is the source directory from which the files must be backed up.
- **Second directory** - This is the destination directory to which the files should be backed up.

In the `rsync` command, if the source or destination directory is preceded by a colon and hostname, it is considered to be a remote directory. The user can backup files from one remote computer to another remote computer. In this case, the user precedes the directory to be backed up with a colon and hostname. The same would be repeated with the destination directory.

An example of rsync command is as follows:

```
rsync -avz /home test1.abc.com:backup_storage/
```

where,

The -a parameter initiates the archive mode that indicates that it keeps the files and directories along with the ownership and permissions of the files.

The -v parameter displays the progress message indicating the sent and received data.

The -z parameter compresses the data being transferred, which helps in saving data transfer time.

The home directory is being copied to a directory called **backup_storage** on a remote server called test1.abc.com.

Figure 13.10 shows an example of files being backed up on the same computer using rsync.

```
[root@localhost /]# rsync -avz /home /backup
sending incremental file list
/home/
/home/john/
/home/john/.bash_history
/home/john/.bash_logout
/home/john/.bash_profile
/home/john/.bashrc
/home/josh/
/home/josh/.bash_logout
/home/josh/.bash_profile
/home/josh/.bashrc

sent 1062 bytes  received 157 bytes  2438.00 bytes/sec
total size is 694  speedup is 0.57
[root@localhost /]# _
```

Figure 13.10: File Backup Using rsync

The output of the rsync command is shown in Figure 13.11.

```
root@localhost /]# cd backup
root@localhost backup]# ls
ome
root@localhost backup]# _
```

Figure 13.11: Output of rsync Command

→ Nautilus

Nautilus is a multipurpose utility that allows connecting to a remote computer. It offers SSH as a service that can be used for connecting to a remote computer. Figure 13.12 shows the Nautilus SSH dialog box.

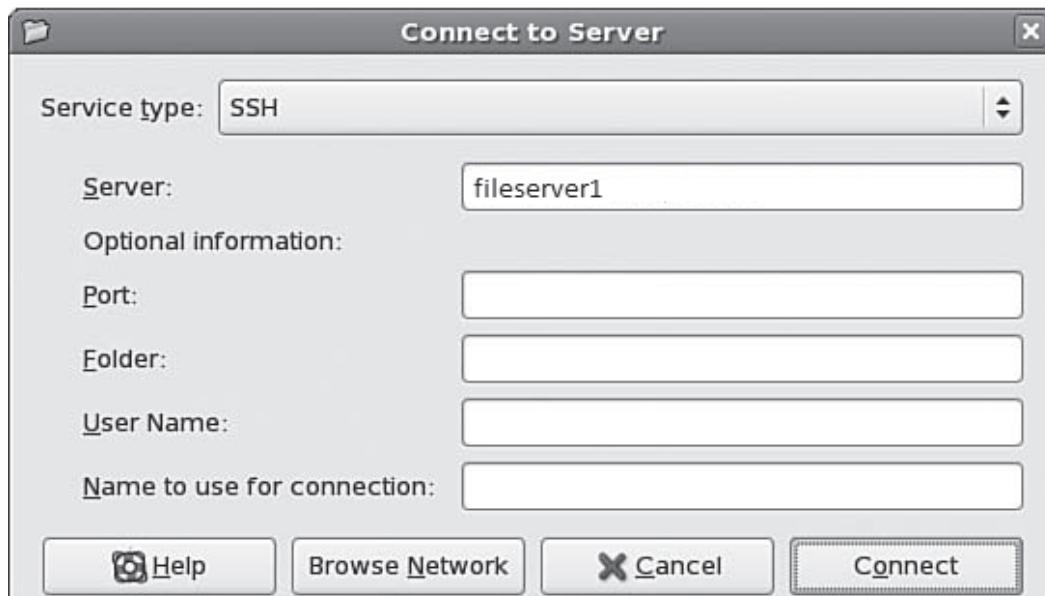


Figure 13.12: Nautilus SSH Dialog Box

There are different parameters that can be defined when initiating a connection. For example, if users are connecting to a Secure File Transfer Server (sftp) server, users should define the port if it is other than port 21. If the users are using the default port 21, the users are not required to define anything.

13.5 Deploying File Sharing Services

RHEL offers FTP as its file sharing services. It also offers other services, such as NFS and Samba.

FTP follows client-server architecture in which one computer acts as the server and other computer acts as the client. The client computer connects to the server computer and initiates a request to either upload or download files. The server computer, which is known as the FTP server, accepts the incoming connections from the sending computer, which is known as the FTP client.

Note - An FTP server can accept anonymous or authenticated connections. If the FTP server is configured to accept anonymous connections, the user is not required to provide any credentials. If the FTP server is configured to accept only authenticated connections, the user should provide the username and password. It is important to note that the username and password are transmitted in clear text, which is prone to interception by the hackers.

The default FTP server for the RHEL operating system is Very Secure FTP Daemon (vsftpd). The key configuration file in which FTP related configuration is stored is `vsftpd.conf`. The file is located in the `/etc/vsftpd` directory.

Users can configure the following parameters in the `vsftpd.conf` file. Some of the key parameters are as follows:

- **listen_port** - Contains the default value as 21, which is the port that FTP server would use. If users are to use any other port, users must change the default value.
- **ftpd_banner** - Defines the banner. This is the welcome message after the user logs on to the FTP server.
- **local_enable** - Contains the default value as NO, which indicates that the local users are not allowed to log on to the FTP server.
- **hide_ids** - Contains the default value as NO.
- **max_clients** - Contains the default value as 0, which indicates unlimited number of connections.

To use FTP, install the vsftpd package from the Red Hat Network, configure the vsftpd.conf file and start the vsftpd service using the service `vsftpd start` command.

RHEL provides a tighter control over the anonymous connections. The vsftpd.conf file provides a number of configurable parameters.

Some of the parameters are as follows:

- anonymous _ enable
- allow _ anon _ ssl
- anon _ mkdir _ write _ enable
- anon _ other _ write _ enable
- anon _ world _ readable _ only
- deny _ email _ enable
- no _ anon _ password

Allowing or Denying User Connections

To allow the user connections, the `user_list` file stored in the `/etc/vsftpd` directory can be used. However, this file is applicable only when users enable the `userlist_enable` directive in the `vsftpd.conf` file. The `userlist_deny` parameter must be set to NO to allow the users in the `userlist` to connect to the FTP server.

Users can use the `ftpusers` file stored in the `/etc/vsftpd` directory to deny any users from connecting to the FTP server. This file contains two users by default, `root` and `nobody`.

Note - The users, who connect to FTP server and are using RHEL, can either use the command line based FTP clients, such as, ftp and lftp. The client can also use the Graphical FTP (gFTP), which is available in **Applications → Internet → gFTP**.

Configuring a Web Server

FTP server follows a client-server model. While the FTP server is meant for uploading and downloading files, the Web server is meant to serve Web pages to the clients. Web server has two key components that are as follows:

- Web server
- Client

RHEL can be configured with Apache Web server that mainly uses two different protocols, TCP and UDP. The default port that is used by HTTP is port 80.

To configure a Web server, users must download and install the httpd package from the Red Hat Network. After the server is installed, users must configure the httpd.conf file that is located in the **/etc/httpd** directory. There are three key sections in this file that are as follows:

- **Global Configuration:** Contains the global configuration values, such as ServerRoot, KeepAlive and Listen.
- **Main Server:** Contains the server specific parameters, such as ServerAdmin, ServerName and DocumentRoot.
- **Virtual Hosts:** Allows users to configure more than one Web site in the form of virtual hosts.

Starting and Stopping the Web Server

To start and stop the Web server, a user must be a root user. The following command must be executed to start the Web server:

```
service httpd start
```

The following command should be executed to stop the Web server:

```
service httpd stop
```

13.6 NFS

NFS is a server-client protocol that allows sharing of files between computers which are located on a common network. NFS is available on a variety of Linux and UNIX platforms. To be able to connect to the NFS server, the client must use a NFS server compatible client application.

NFS is available in different versions. RHEL supports different versions that are as follows:

- NFSv2
- NFSv3
- NFSv4

By default, RHEL uses NFSv4, which has the capability to work through firewalls. It also supports Access Control Lists (ACLs) and stateful operations. It requires Transmission Control Protocol (TCP) that is running over the IP network. The previous versions, NFSv2 and NFSv3, use User Datagram Protocol (UDP) over an IP network. TCP port 2049 is the default port that is used by NFSv4.

13.6.1 Basics of NFS

NFS offers a number of advantages. It makes file sharing simple and effortless. Following are the advantages:

- The server and client can use different operating systems.
- The client mounts the remote directories to local directories.
- Access is granted to the client IP address and therefore, no user authentication is required.

Following steps are followed to connect NFS client and server:

1. The NFS server exports a directory to the client system as requested.
2. The client system mounts the directory to a local directory. This is called a mount point.
3. The Input/Output (I/O) operations are written to the server.
4. Client recognizes the changes as if they are being performed on the local directories, rather than the remote directories.

In this process, local directories are not refreshed by the client because remote Filesystem is mapped to local directories.

13.7 Configure NFS

To be able to use the NFS shares on the server, the client must mount the shares first. Following command is used to mount the shares:

`mount`

An example of this command is as follows:

```
mount -t nfs server://remote_directory/local_directory
```

where

`remote_directory` is the remote host directory that should be mounted.

`local_directory` is the local directory where remote directory should be mounted.

By default, NFSv4 uses the `mount` command with `-t nfs` parameters. When a server does not support NFSv4, the client automatically uses the version that is supported by the server. It is important to note that an NFS share that is manually mounted is no longer available after the client reboots.

Mounting NFS

Editing and adding a line in the `/etc/fstab` file is a method to mount an NFS share.

The user must be a root user to modify this file.

Following syntax is used to add a line to the `/etc/fstab` file:

```
server:/remote/export /local/directory nfs options 0 0
```

Following parameters must be contained in the command:

- ➔ Hostname of the NFS share
- ➔ Directory that is being exported from the server
- ➔ Directory on which share is being mounted on the local machine

One of the disadvantages of using the `/etc/fstab` file is that it reserves the resources for the mounts. This implies that if a server has a large number of mounts, there should be enough resources available for mounts to work.

13.8 AutoFs Service

The `autofs` service is used to monitor preconfigured NFS mount points. The `autofs` service uses the `automount` daemon for this.

The `automount` utility is an alternative to modifying the `/etc/fstab` file. There are two components to the `automount` utility, which consists of two components.

Following are the components:

- ➔ **Kernel Module** - Implements a file system.
- ➔ **User-space Daemon** - Performs different functions.

The automount utility performs on-demand mounting, which refers to mounting and unmounting file systems automatically. Along with the NFS, automount can also be used for mounting different file systems.

Following are the file systems that are supported by automount:

- ➔ Andrew File System (AFS)
- ➔ Samba File System (SMBFS)
- ➔ Common Internet File System (CIFS)
- ➔ Local File Systems

The nfs-utils package should be installed on the RHEL system before using the automount utility.

The autofs service uses the **/etc/auto.master** file as the default configuration file. Figure 13.13 displays the sample **auto.master** file.

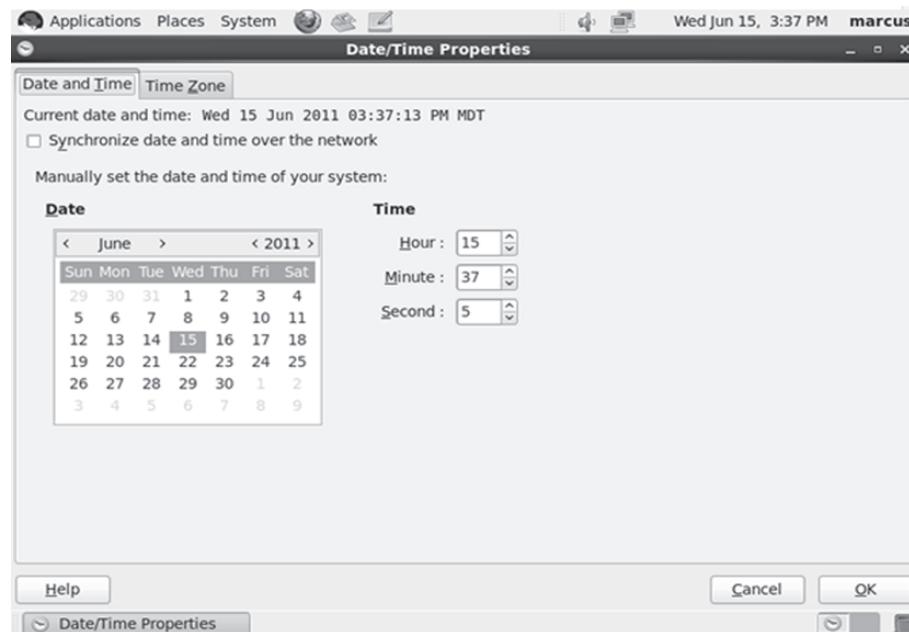


Figure 13.13: Sample auto.master File

The auto.master file uses a redirection for the mounts to the **/misc/** directory. The redirection is mapped to the **/etc/auto.misc** file. Figure 13.14 shows the auto.misc redirection.

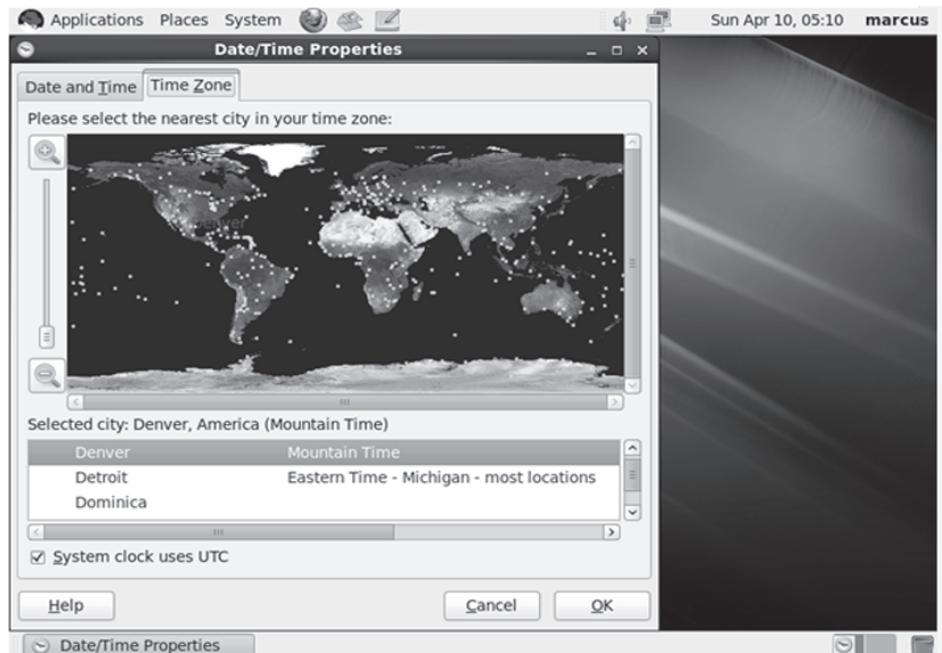


Figure 13.14: auto.misc Redirection

The **/etc/auto.misc** file is displayed in Figure 13.15.

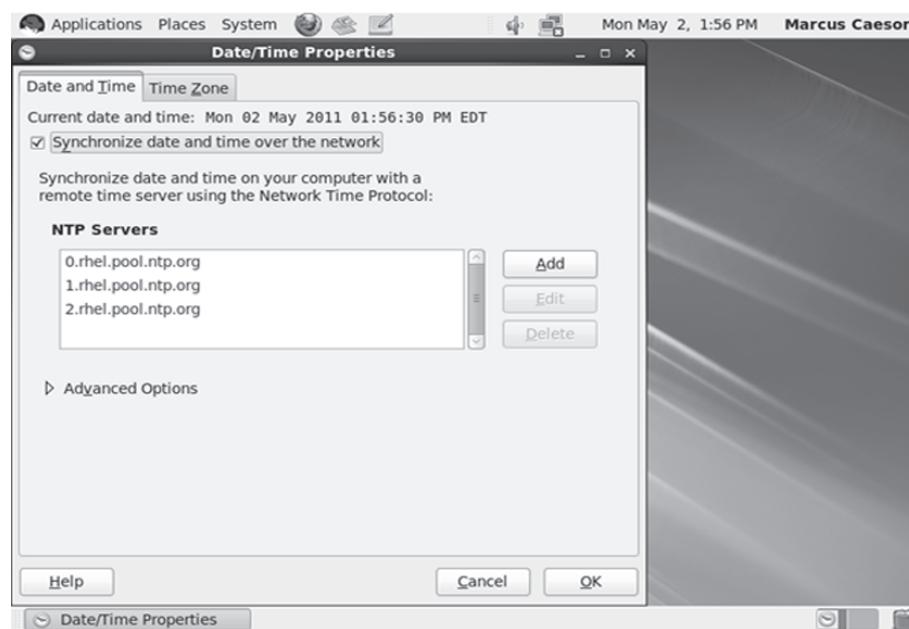


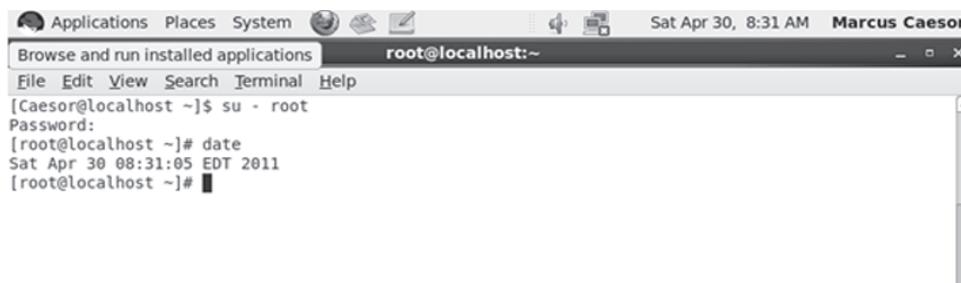
Figure 13.15: /etc/auto.misc File

Starting the automount Daemon

To start the automount daemon, the system administrator must execute the following command:

```
service autofs start
```

Figure 13.16 shows an attempt to start the autofs service.



A screenshot of a terminal window titled 'root@localhost:~'. The window shows a root shell session. The user runs the command 'su - root' to become root. Then, they run 'date' to check the current date and time. Finally, they attempt to start the 'autofs' service with the command 'service autofs start'. The terminal window has a standard Gnome-style interface with a menu bar and icons at the top.

```
[Caesar@localhost ~]$ su - root
Password:
[root@localhost ~]# date
Sat Apr 30 08:31:05 EDT 2011
[root@localhost ~]# [root@localhost ~]# service autofs start
[root@localhost ~]#
```

Figure 13.16: Starting the autofs Service

Note that if the service is already running, RHEL automatically flags that the service is running along with its status.

To stop the automount daemon, the system administrator must use the command as follows:

```
service autofs stop
```

Figure 13.17 shows stopping the autofs service.



A screenshot of a terminal window titled 'root@localhost:~'. The window shows a root shell session. The user runs the command 'su - root' to become root. Then, they run 'date' to check the current date and time. After that, they run 'date -s "1 MAY 2011 18:00:00"' to change the system date to May 1, 2011, at 18:00:00. Finally, they stop the 'autofs' service with the command 'service autofs stop'. The terminal window has a standard Gnome-style interface with a menu bar and icons at the top.

```
[Caesar@localhost ~]$ su - root
Password:
[root@localhost ~]# date
Sat Apr 30 08:31:05 EDT 2011
[root@localhost ~]# date -s "1 MAY 2011 18:00:00"
Sun May 1 18:00:00 EDT 2011
[root@localhost ~]# [root@localhost ~]# service autofs stop
[root@localhost ~]#
```

Figure 13.17: Stopping the autofs Service

Note - The user must have root user privileges to start or stop the services in RHEL.

Exercise - 1

→ Viewing the `auto.master` file

Note - User must have root permissions to edit the `rsyslog.conf` file. All other users should have Read permissions.

Step 1 - Log on to the RHEL workstation using root username and password.

Step 2 - On the desktop, double-click Computer (Refer to Figure 13.18).

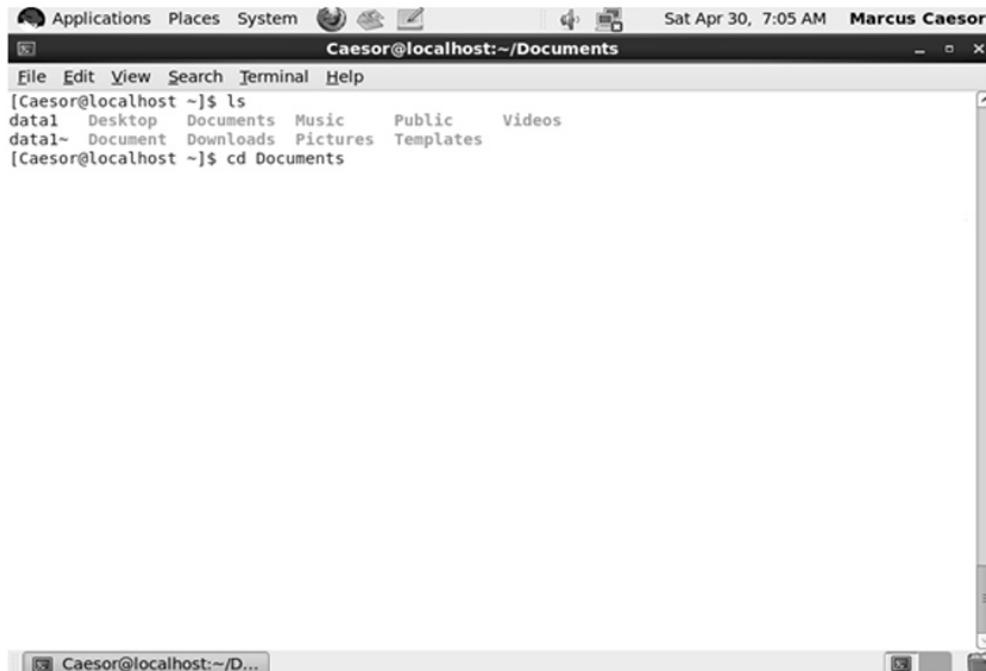
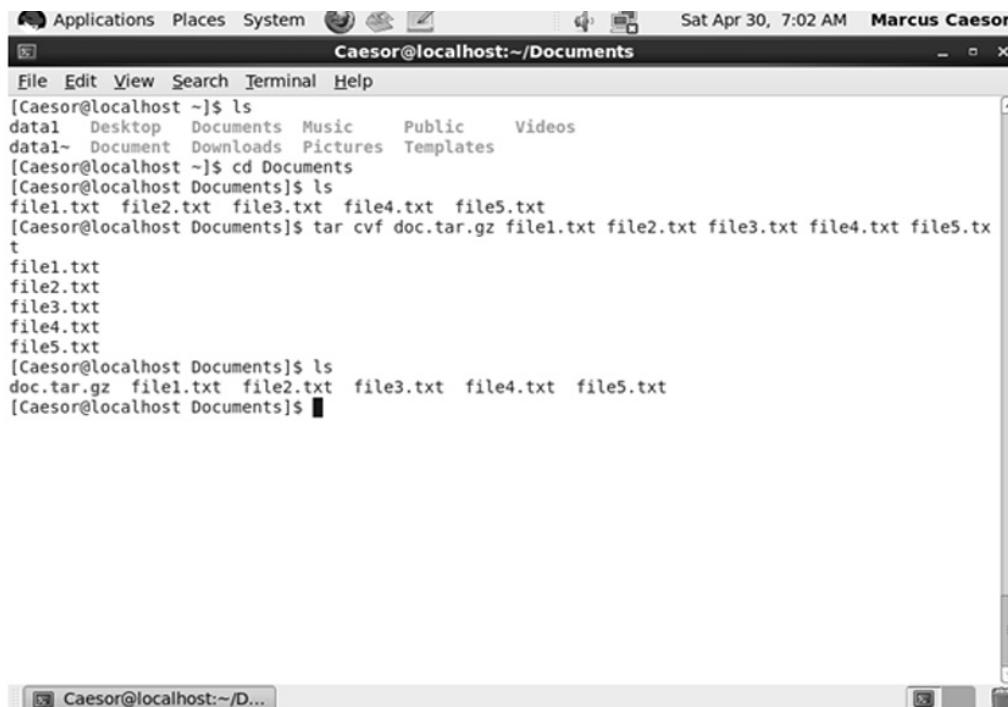


Figure 13.18: Computer Icon

The contents of the computer are displayed in Figure 13.19.



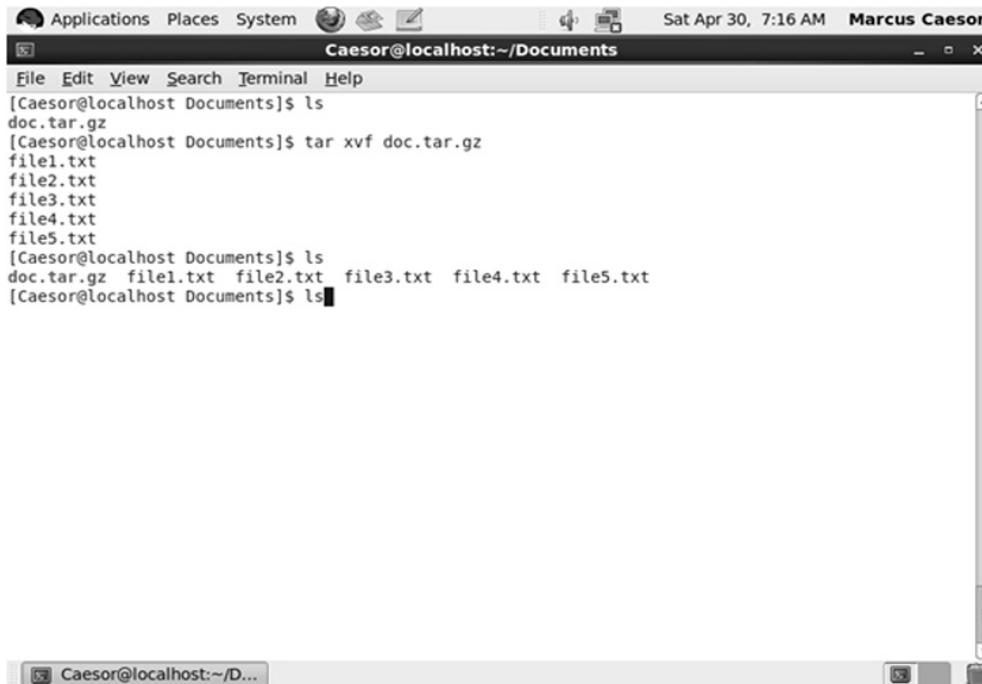
A screenshot of a terminal window titled "Caesor@localhost:~/Documents". The window shows a command-line session where the user is creating a tar archive named "doc.tar.gz" containing files "file1.txt", "file2.txt", "file3.txt", "file4.txt", and "file5.txt". The session starts with an "ls" command showing directories "Desktop", "Documents", "Music", "Public", and "Videos", and files "data1", "data1~", "Document", "Downloads", "Pictures", and "Templates". The user then changes directory to "Documents" and lists its contents again, which now include the five created text files. Finally, the user runs the "tar cvf doc.tar.gz file1.txt file2.txt file3.txt file4.txt file5.txt" command to create the archive.

```
[Caesor@localhost ~]$ ls
data1 Desktop Documents Music Public Videos
data1~ Document Downloads Pictures Templates
[Caesor@localhost ~]$ cd Documents
[Caesor@localhost Documents]$ ls
file1.txt file2.txt file3.txt file4.txt file5.txt
[Caesor@localhost Documents]$ tar cvf doc.tar.gz file1.txt file2.txt file3.txt file4.txt file5.txt
t
file1.txt
file2.txt
file3.txt
file4.txt
file5.txt
[Caesor@localhost Documents]$ ls
doc.tar.gz file1.txt file2.txt file3.txt file4.txt file5.txt
[Caesor@localhost Documents]$
```

Figure 13.19: Computer Contents

Step 3 - Double-click Filesystem and open the **/etc/ directory** (Refer to Figure 13.20).

Note - There are a large number of files and directories in the **/etc/ directory**. A user can scroll down to locate the required files or directories.

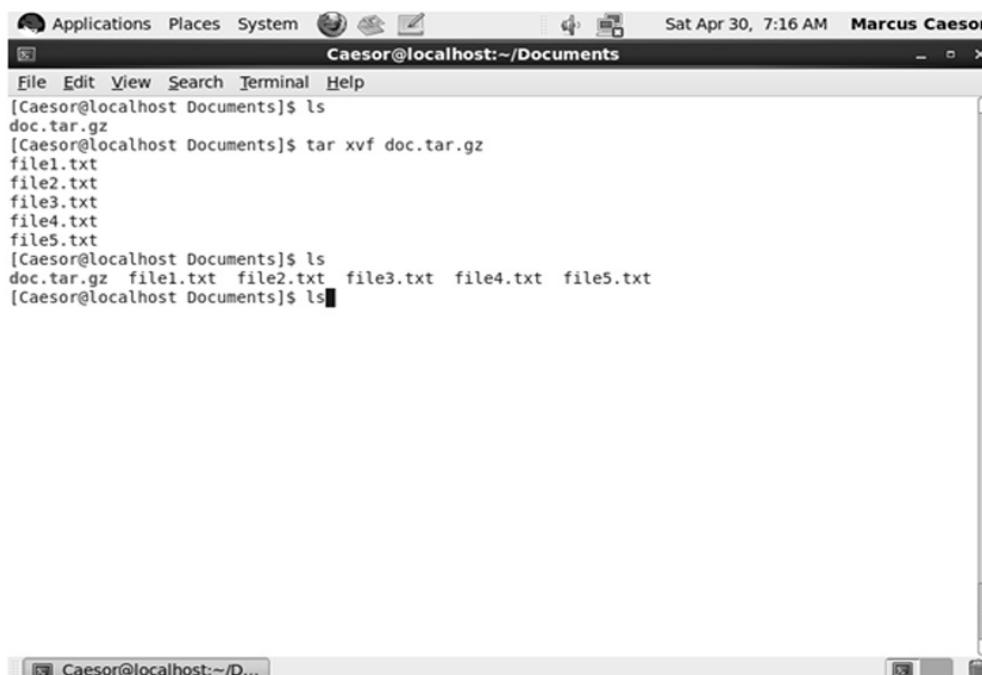


A screenshot of a Linux desktop environment showing a terminal window. The window title is "Caesor@localhost:~/Documents". The terminal shows the following command sequence:

```
[Caesor@localhost Documents]$ ls  
doc.tar.gz  
[Caesor@localhost Documents]$ tar xvf doc.tar.gz  
file1.txt  
file2.txt  
file3.txt  
file4.txt  
file5.txt  
[Caesor@localhost Documents]$ ls  
doc.tar.gz file1.txt file2.txt file3.txt file4.txt file5.txt  
[Caesor@localhost Documents]$ ls
```

Figure 13.20: /etc/ Directory

Step 4 - Double-click **auto.master** to open the file as shown in Figure 13.21.



A screenshot of a Linux desktop environment showing a terminal window. The window title is "Caesor@localhost:~/Documents". The terminal shows the same command sequence as Figure 13.20:

```
[Caesor@localhost Documents]$ ls  
doc.tar.gz  
[Caesor@localhost Documents]$ tar xvf doc.tar.gz  
file1.txt  
file2.txt  
file3.txt  
file4.txt  
file5.txt  
[Caesor@localhost Documents]$ ls  
doc.tar.gz file1.txt file2.txt file3.txt file4.txt file5.txt  
[Caesor@localhost Documents]$ ls
```

Figure 13.21: Contents of the auto.master File

Study the contents of the file and notice there is a **/misc** directory mapping in this file.

Step 5 - Close the auto.master file.

Step 6 - In the **/etc/** directory, locate and open the auto.misc file.

Figure 13.22 shows the contents of the auto.misc file.



Figure 13.22: Contents of auto.misc File

Notice that there is mapping of cd-rom marked with the following statement:

```
cd -fstype=iso9660, ro, nosuid, nodev : /dev/cdrom
```

Step 7 - Review the file contents and close the file.

13.9 Check Your Progress

1. What is the minimum permission required to change the date and time?

(A)	Root	(C)	System Admin
(B)	Daemon	(D)	User

2. State true or false:

It is possible to add internal time server for time synchronisation.

(A)	True	(C)	False
-----	------	-----	-------

3. Which of the following directories maintain the system log?

(A)	/etc/log	(C)	/var/log
(B)	/etc/var	(D)	/var/sys/log

4. Which of the two utilities are required to archive and compress files?

(A)	tar, gzip	(C)	tar, vi
(B)	tar, winzip	(D)	tar, arch

5. Which of the following commands are used to browse the Web from the command prompt?

(A)	links Web <Website name>	(C)	<Website name>
(B)	elinks <Website name>	(D)	links <Website name>

6. Which network configuration files are used to configure the network connection for a network adapter?

(A)	Root	(C)	System Admin
(B)	Daemon	(D)	User

7. What is the correct command to edit the network adapter configuration file, considering there is a single network adapter installed in RHEL computer?

(A)	True	(C)	False
-----	------	-----	-------

8. Which of the following is the correct command for starting the ssh service?

(A)	/etc/log	(C)	/var/log
(B)	/etc/var	(D)	/var/sys/log

9. The httpd.conf file is divided into how many sections?

(A)	tar, gzip	(C)	tar, vi
(B)	tar, winzip	(D)	tar, arch

10. What is the default value of listen_port in the vsftpd.conf file?

(A)	links Web <Website name>	(C)	<Website name>
(B)	elinks <Website name>	(D)	links <Website name>

11. Which package must be installed for automount utility?

(A)	nfs-utils	(C)	automount-utils
(B)	nfs	(D)	amount

12. Which file contains the redirection for the /etc/auto.misc file?

(A)	auto.conf	(C)	master.conf
(B)	auto.master	(D)	misc.master

13. Which command is used to add an NFS share on a client?

(A)	mount	(C)	nfs add
(B)	add	(D)	cat

14. Which directory stores the auto.master file?

(A)	/etc/	(C)	/usr/
(B)	/var/	(D)	/bin/

15. Which directory contains the fstab file?

(A)	/etc/	(C)	/usr/
(B)	/var/	(D)	/bin/

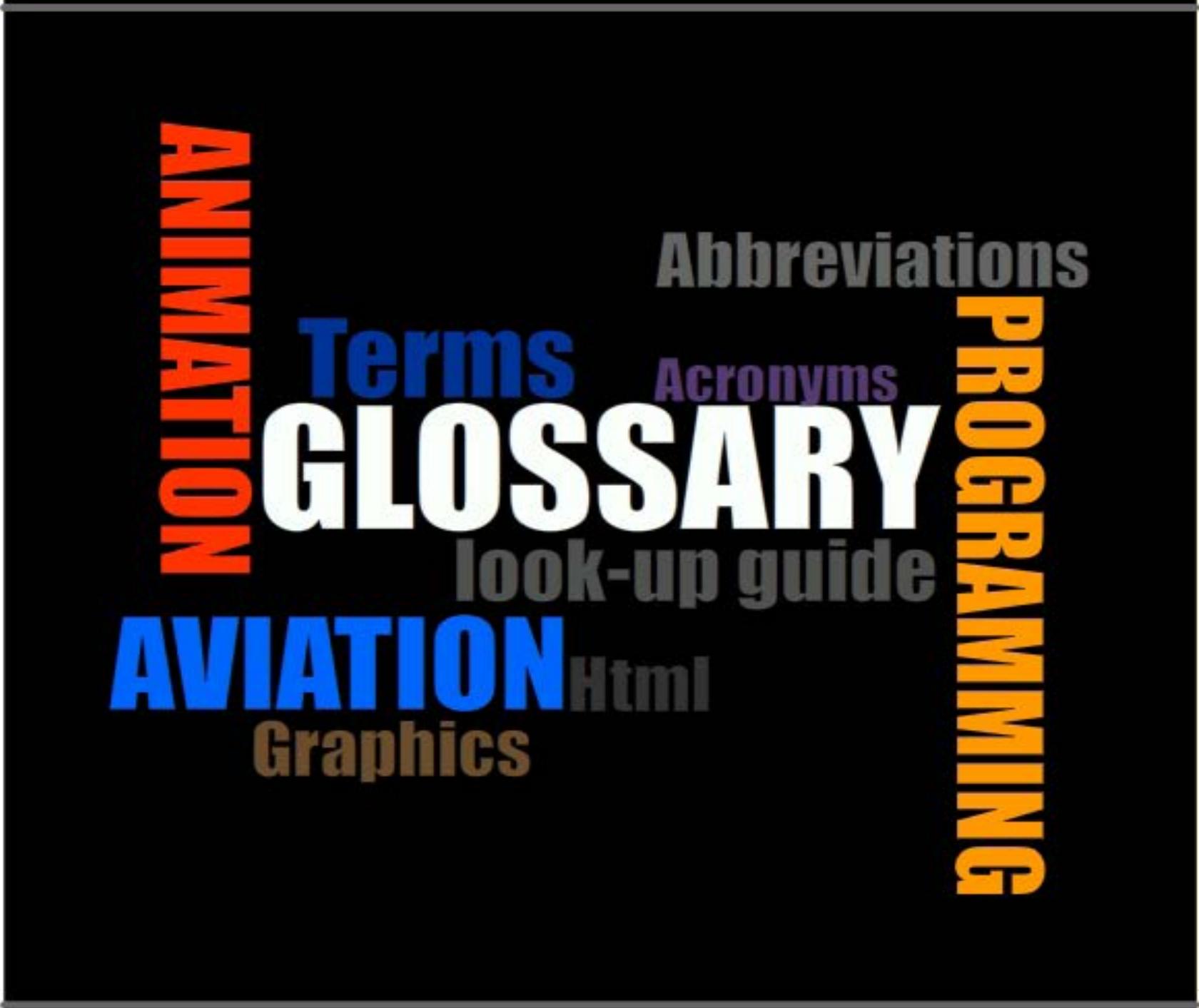
13.9.1 Answers

1.	A
2.	A
3.	C
4.	A
5.	D
6.	A
7.	D
8.	D
9.	C
10.	B
11.	A
12.	B
13.	A
14.	A
15.	A



Summary

- The three different methods of configuring network in RHEL are through the command line, through the graphical interface in X Window System and by configuring the configuration files. There are three different methods of remotely administering remote system namely, by using Rsync, Secure Shell (SSH), and Nautilus.
- RHEL can be configured as a FTP server that can be configured to accept anonymous connections or to deny access to users. RHEL can also be configured as a Web server. By default, Apache is the Web server that can be configured on RHEL.
- There are different variants of Network File System (NFS). These are NFSv2, NFSv3, and NFSv4. NFSv4 communicates over TCP and uses port 2049. For NFS communication to take place, a client should mount the shares first. The mount command is used to mount the shares.
- NFS mount can be done using two methods. The first method is by using the /etc/fstab file and second method is the automount utility. The autofs service uses the automount daemon. It uses this daemon to monitor the preconfigured NFS mount points. The autofs service uses the /etc/auto.master file as the default configuration file.



ANIMATION

AVIATION

TERMS

GLOSSARY

look-up guide

ACRONYMS

HTML

GRAPHICS

PROGRAMMING

Abbreviations

Exercise - 1

Configuring a printer

Step 1 - To add a new printer, open Printer Configuration, from **System → Administration → Printing**.

The Printer configuration - localhost window, as shown in Figure 14.1.

Note - For command line-based environment, execute the command system-config-printer at the command prompt.



Figure 14.1: Printer configuration - localhost Window

Step 2 - Click New.

Step 3 - In the Authenticate dialog box (Refer to Figure 14.2), specify the root password and click Authenticate.



Figure 14.2: Authenticate Dialog Box

Step 4 - Select the type of printer to be installed. A user can choose local printer or configure a printer from the network as shown in Figure 14.3. Click Serial Port #1 and click Forward.



Figure 14.3: Selecting a Printer

Step 5 - In the New Printer dialog box (Refer to Figure 14.4) in the Makes list, scroll down to select HP from the list and click Forward.



Figure 14.4: New Printer Dialog Box

Step 6 - In the Choose Driver page (Refer to Figure 14.5), click Forward.

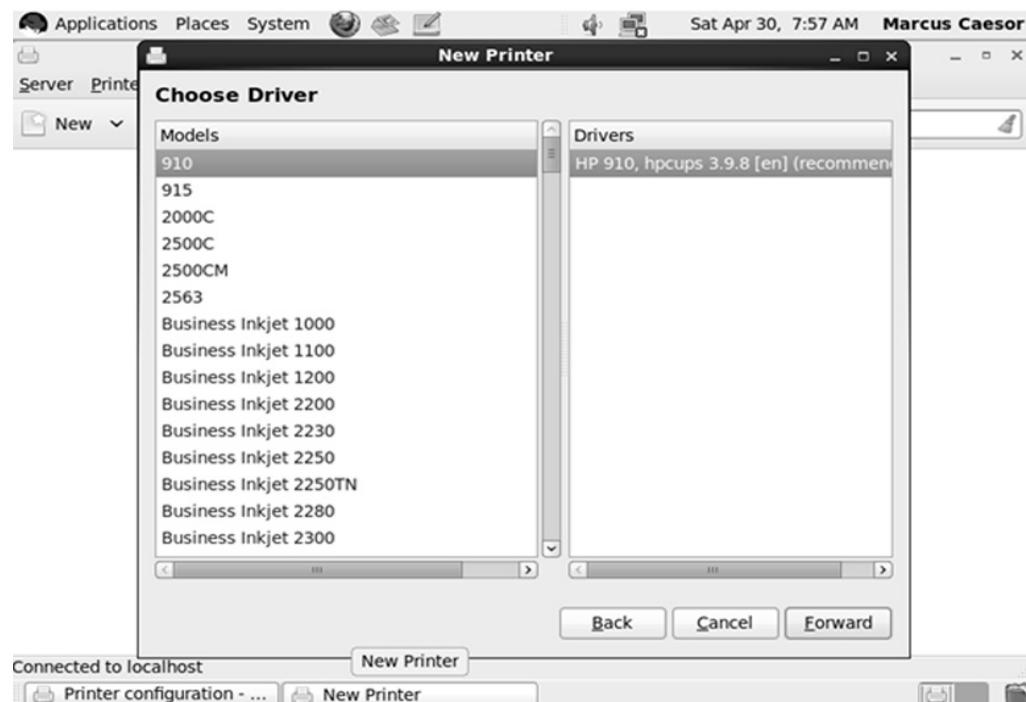


Figure 14.5: Choose Driver Page

Step 7 - In the Installable Options page (Refer to Figure 14.6), click Forward.



Figure 14.6: Installable Options Page

Step 8 - On the Describe Printer page (Refer to Figure 14.7), provide the Printer Name, Description and Location and then click Apply.



Figure 14.7: Describe Printer Page

Step 9 - Specify the root password and click Authenticate.

Step 10 - Confirm the authentication by specifying the root password, and then click OK.

Step 11 - The printer is now installed. A prompt is displayed to print a test page as shown in Figure 14.8. Click No.



Figure 14.8: Prompt to Print a Test Page

The printer is now successfully installed as shown in Figure 14.9.



Figure 14.9: Installed Printer

Step 12 - Close the Printer configuration - localhost dialog box.

Exercise - 2**Configuring network at the command prompt**

Note - This method is used when a user does not have X Window System installed on the RHEL system. The user can also be an experienced administrator who prefers to work on command prompt.

Step 1 - Execute the following command at the shell prompt to invoke the network configuration utility.

```
# system-config-network
```

Step 2 - On the Select Action screen (Refer to Figure 14.10), Device configuration is selected by default.

Press ENTER.



Figure 14.10: Select Action Screen

Step 3 - In the Select A Device screen (Refer to Figure 14.11) all the installed network adapters would be listed. The first adapter would be marked as eth0 along with the brand and make of the adapter. Press ENTER to configure the network configuration properties for this network adapter.



Figure 14.11: Select A Device Screen

Step 4 - On the Network Configuration screen (Refer to Figure 14.12), navigate to Use DHCP and press SPACEBAR. This enables the Static IP, Netmask and Default gateway IP fields.

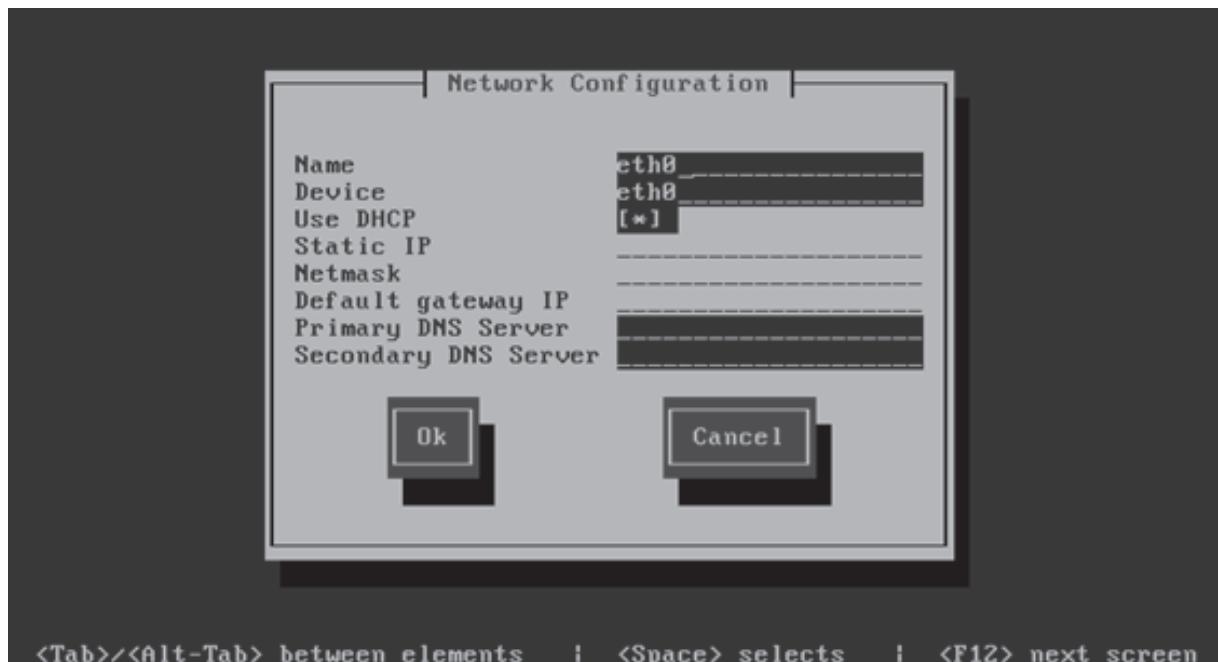


Figure 14.12: Network Configuration Screen

Note - By default, the network adapter would be configured to use Dynamic Host Configuration Protocol (DHCP) server. However, if it is a server that is going to be hosted on a network to provide services, such as Web or FTP, to the users, it is better to configure static IP. If a system is configured to use DHCP, the user is not required to define the Primary DNS Server or Secondary DNS Server IP address. Both these IP addresses would be assigned by the DHCP server.

Step 5 - Type the following information for the respective fields (Refer to Figure 14.13):

Static IP: 192.168.10.100

Netmask: 255.255.255.0

Default gateway IP: 192.168.10.1

Primary DNS Server: 192.168.10.1

Secondary DNS Server: 192.168.10.2

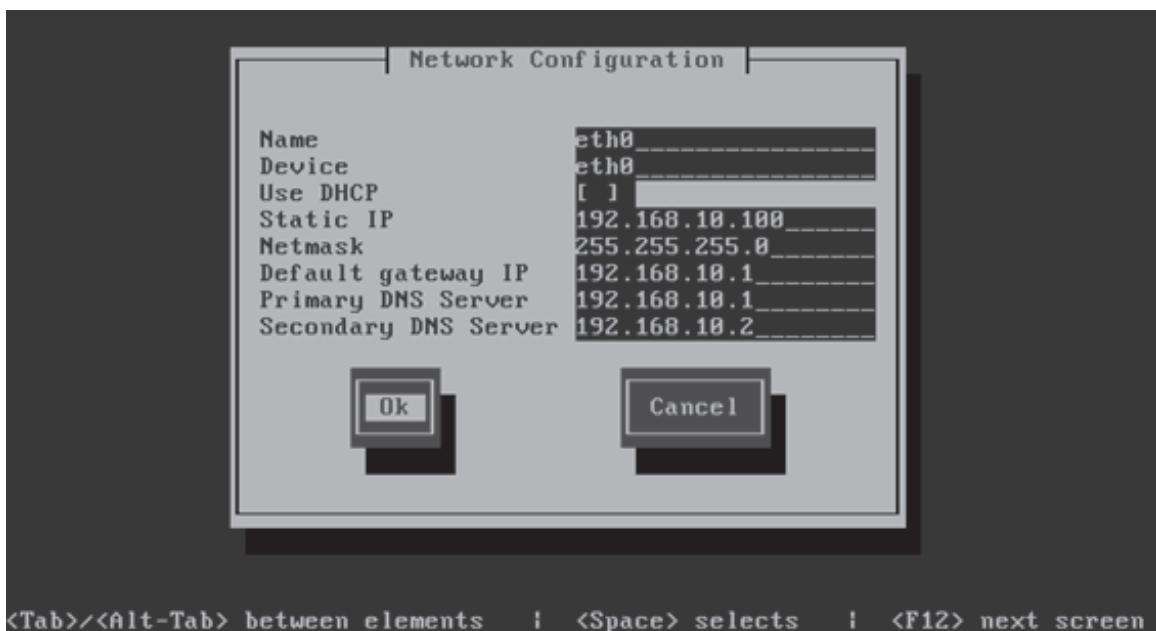


Figure 14.13: Network Configuration Information

Step 6 - Select Ok and press ENTER. This goes back to the Select A Device screen.

Step 7 - Select Save and press ENTER.

Step 8 - On the next screen, select Save&Quit and press ENTER to save the network configuration for the network adapter.

Note - Restart the network services after making changes to the network configuration. To implement the changes, it is necessary to restart the network services.

Exercise - 3

Configuring network in X Window System graphical environment

Note - A user can also run the system-config-network tool in the graphical environment. Both the command line and graphical version provide the same end result.

Step 1 - Execute the following command at the shell prompt to invoke the network configuration utility.

```
# system-config-network &
```

The network adapter configuration would be displayed. All configuration settings that the user had done using the command line utility, are saved and shown in the GUI-based network configuration utility (Refer to Figure 14.14).

Step 2 - Select the Ethernet card (eth0 or eth1) and click Edit, as shown in Figure 14.14.

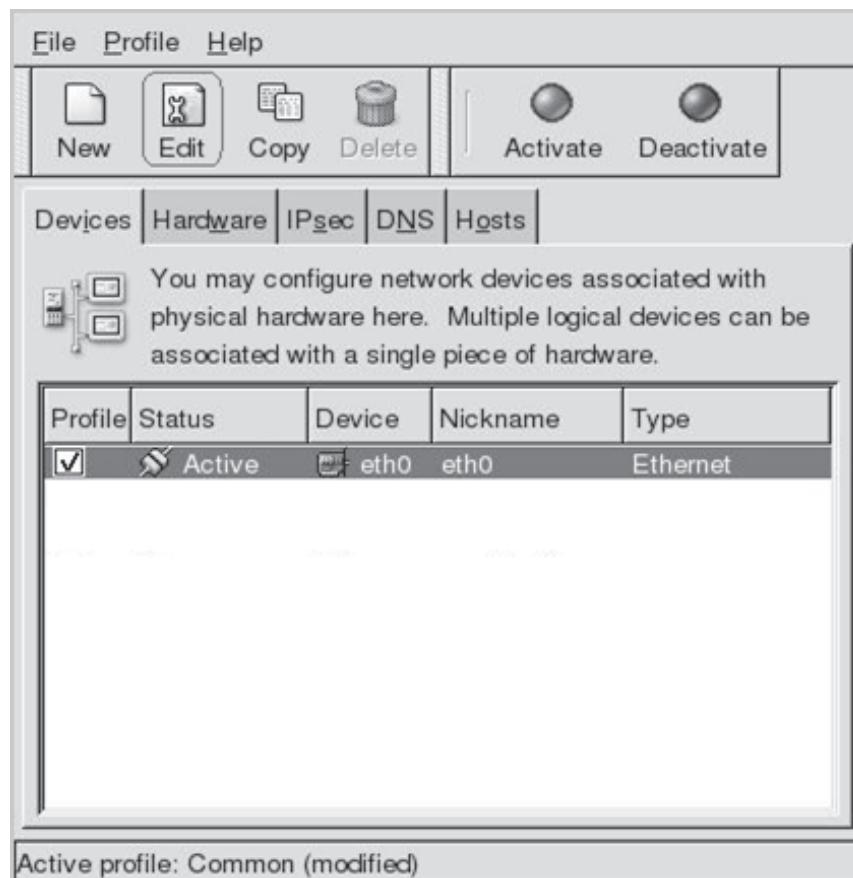


Figure 14.14: Network Configuration Utility

Step 3 - In the Ethernet card properties (Refer to Figure 14.15); provide the IP address, netmask, default gateway and other properties. Click OK.

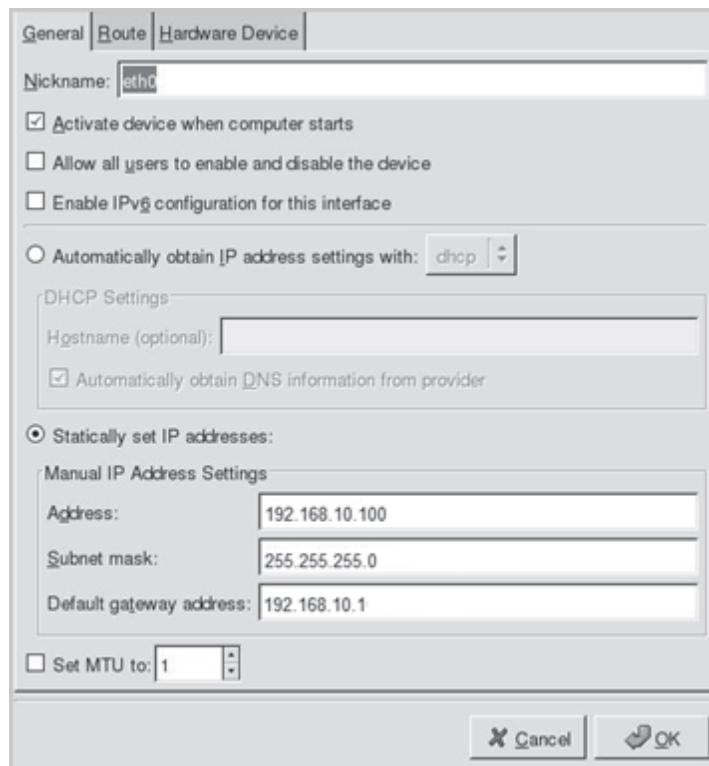


Figure 14.15: Ethernet Card Properties

If the user is familiar with working in graphical mode, the user would find the network configuration utility to be easy to use. All the command line tasks can be performed in the graphical mode with much ease. However, graphical mode would not be available if the user does not have X Window System installed. The graphical network configuration tool can be invoked from **System-> Administration -> Network**.

The network adapter drivers are defined as kernel modules. When RHEL is installed, if user chooses to enable the networking feature, the network adapter drivers or kernel modules for networking are loaded. The kernel modules for networking are defined as alias line in a file named modprobe.conf, which is located in the /etc directory. Depending on the number of network adapters installed, the user would identify an alias line for each of the network adapters. For example, the content that the modprobe.conf file would contain if there are two network adapters installed is as follows:

```
alias eth0 3c59x
alias eth1 e1000
```

Exercise - 4**Configuring network configuration files**

Note - This method allows users to configure network card by making changes to the configuration files that are stored in the /etc/sysconfig/network-scripts/ directory. To make changes to these files, the users must navigate to the /etc/sysconfig/network-scripts/ directory.

Step 1 - Set the active directory to the /etc/sysconfig/network-scripts/ directory by executing the following command at the shell prompt:

```
Cd /etc/sysconfig/network-scripts/
```

Step 2 - Modify the first network adapter configuration file, /etc/sysconfig/network-scripts/ifcfg-eth0 by typing vi ifcfg-eth0. The contents of the ifcfg-eth0 file are shown in Figure 14.16.

Note - If it is the first Ethernet adapter, the users are required to modify the file that is eth0. If there are more Ethernet adapters, the filename would change accordingly, such as eth1 or eth2.

```
DEVICE="eth0"
HWADDR="00:0C:29:E3:57:9F"
DHCPCLASS=
IPADDR=192.168.121.2
NETMASK=255.255.255.0
NM_CONTROLLED="yes"
ONBOOT="no"
```

Figure 14.16: Contents of the ifcfg-eth0 File

Note - When a user executes the vi ifcfg-eth0 command, if the ifcfg-eth0 file does not exist, it would be automatically created. The user must append the necessary parameters to configure the eth0 network adapter.

Step 3 - Define the relevant hostname and gateway in the network file located in the /etc/sysconfig/ <directory> to complete the network configuration.

```
# vi /etc/sysconfig/network
```

Append/modify configuration as follows:

```
NETWORKING=yes
```

```
HOSTNAME=www.example.com
```

GATEWAY=192.168.10.1

Note - Define the hostname and gateway after saving and closing the ifcfg-eth0 file.

Step 4 - Save the file and then close it.

Step 5 - Restart the services for changes to take effect. To restart, execute the following command:

```
# /etc/init.d/network restart
```

Step 6 - Edit the resolv.conf file by executing the following command:

```
# vi /etc/resolv.conf
```

Step 7 - Configure the DNS settings in the resolv.conf file. The following information must be specified:

nameserver 192.168.10.1

nameserver 192.168.10.2

Step 8 - Save and close the file.

Step 9 - Test the connectivity with the computers that exist on the network. Ping a computer with the IP address 192.168.10.10. The following command must be executed to test the network connectivity as shown in Figure 14.17.

```
[root@Marcus network-scripts]# ping 192.168.121.2
PING 192.168.121.2 (192.168.121.2) 56(84) bytes of data.
64 bytes from 192.168.121.2: icmp_seq=1 ttl=128 time=0.528 ms
64 bytes from 192.168.121.2: icmp_seq=2 ttl=128 time=0.659 ms
64 bytes from 192.168.121.2: icmp_seq=3 ttl=128 time=0.498 ms
64 bytes from 192.168.121.2: icmp_seq=4 ttl=128 time=0.548 ms
64 bytes from 192.168.121.2: icmp_seq=5 ttl=128 time=0.536 ms
64 bytes from 192.168.121.2: icmp_seq=6 ttl=128 time=0.458 ms
^C
--- 192.168.121.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5197ms
rtt min/avg/max/mdev = 0.458/0.537/0.659/0.068 ms
[root@Marcus network-scripts]#
```

Figure 14.17: Testing Network Connectivity

Exercise - 5

Configuring IP address using the GUI tool

Step 1 - Click System -> Preferences -> Network Connections.

Figure 14.18 shows the Network Connections dialog box.

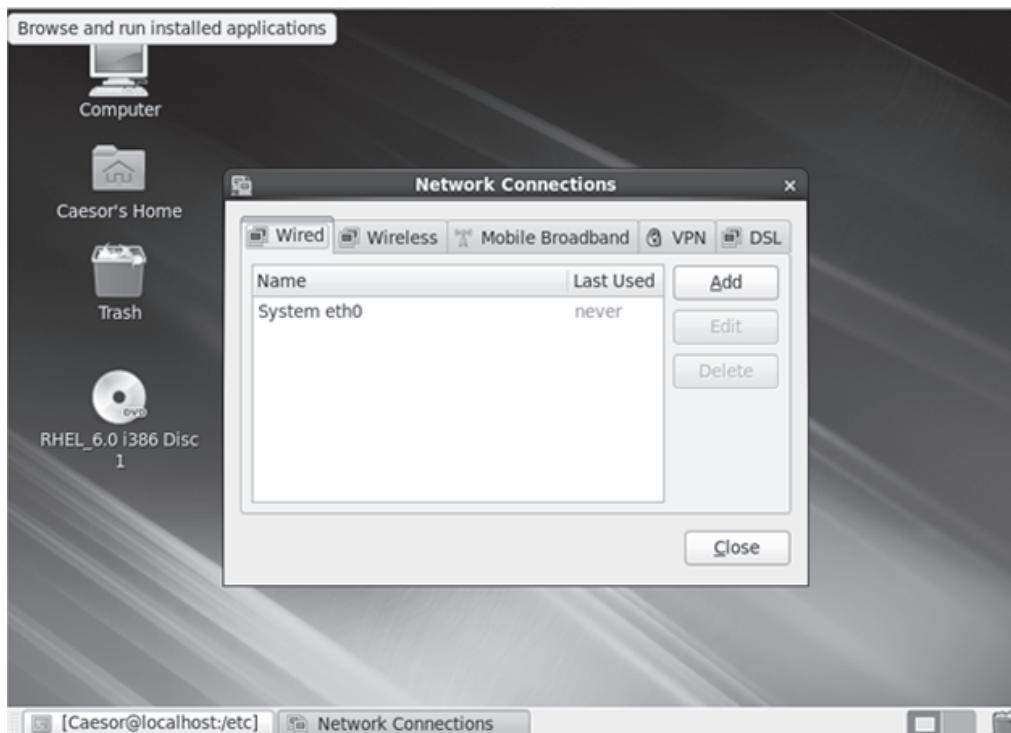


Figure 14.18: Network Connections Dialog Box

Step 2 - Click System eth0 and click Edit.

Figure 14.19 shows the Editing System eth0 dialog box.

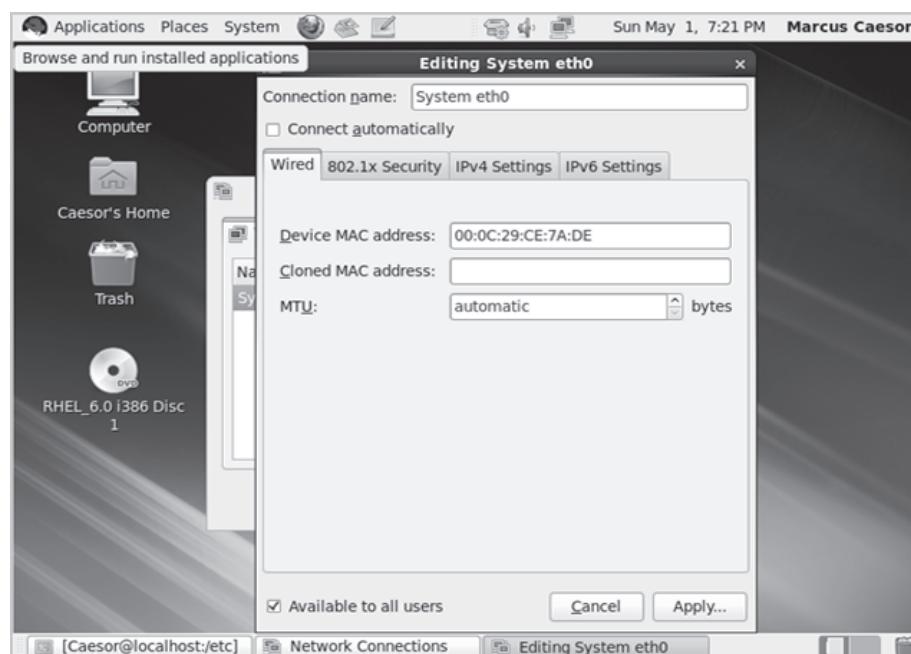


Figure 14.19: Editing System eth0 Dialog Box

Step 3 - Click the IPv4 Settings tab.

Step 4 - From the Method list, select Manual as shown in Figure 14.20.

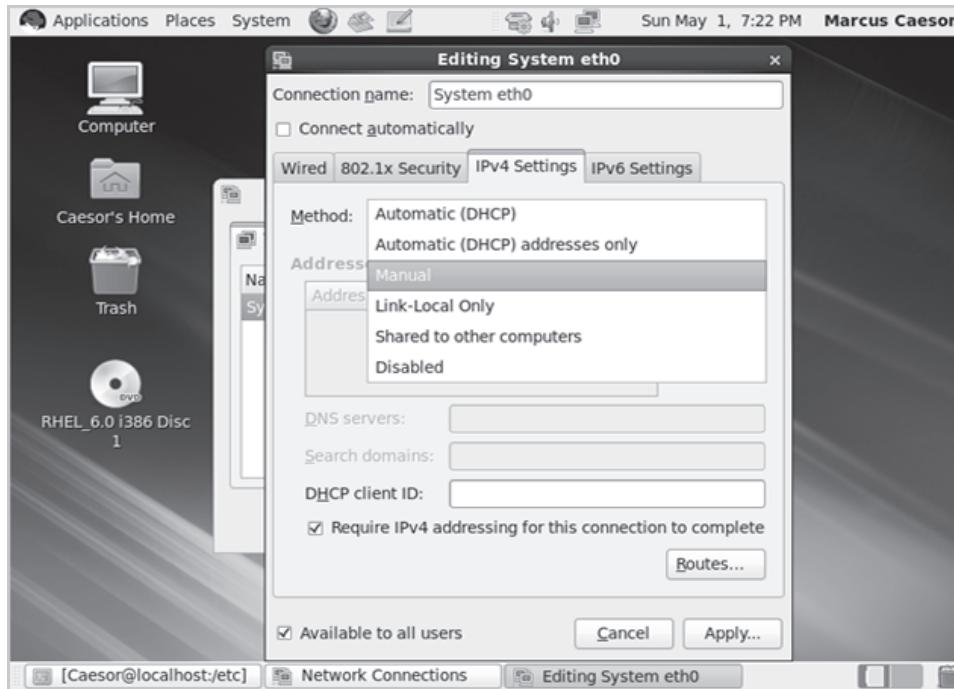


Figure 14.20: Selecting Manual Method

Step 5 - Click Add. An IP address automatically appears. Figure 14.21 displays the manually defined IP address.

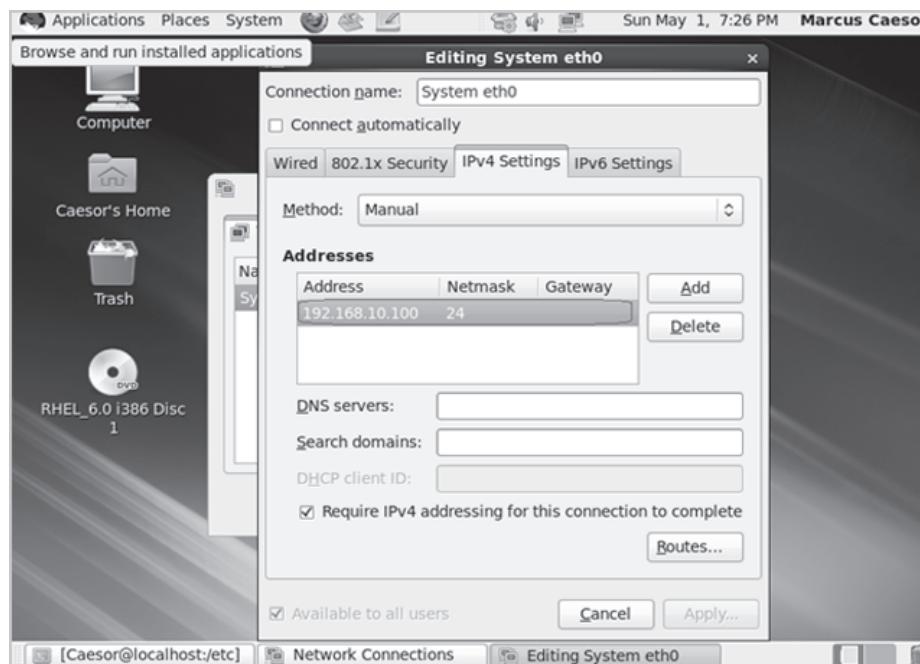


Figure 14.21: Manually Defined IP Address

Step 6 - Click under the Gateway column, to define the gateway address, as shown in Figure 14.22.

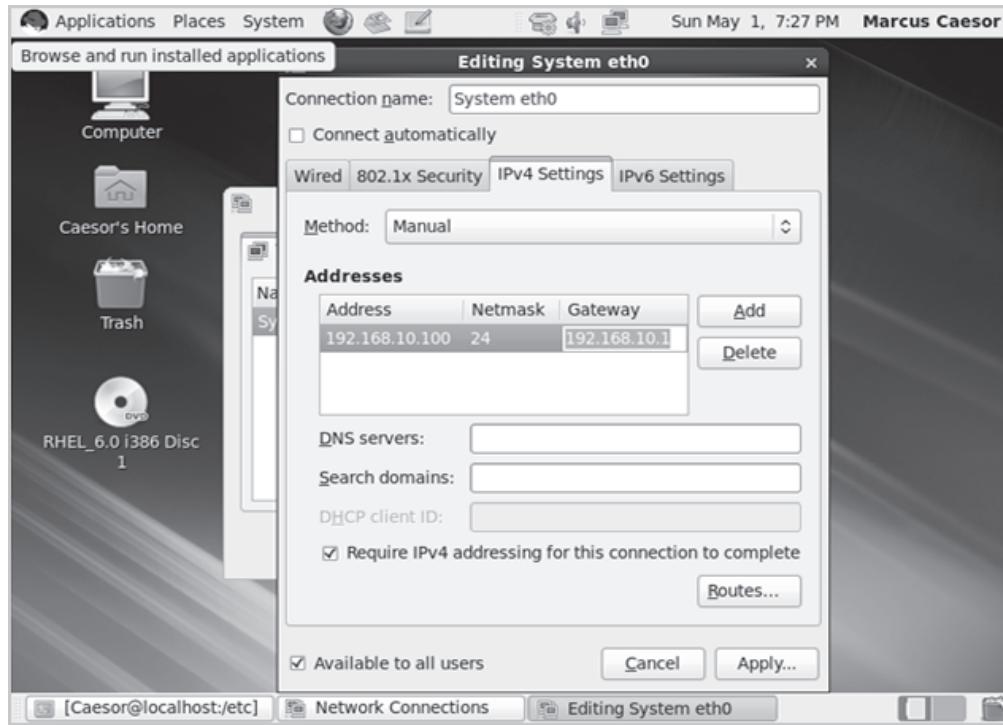


Figure 14.22: Gateway Address

Step 7 - In the DNS server box, type the IP address for the DNS servers as shown in Figure 14.23.

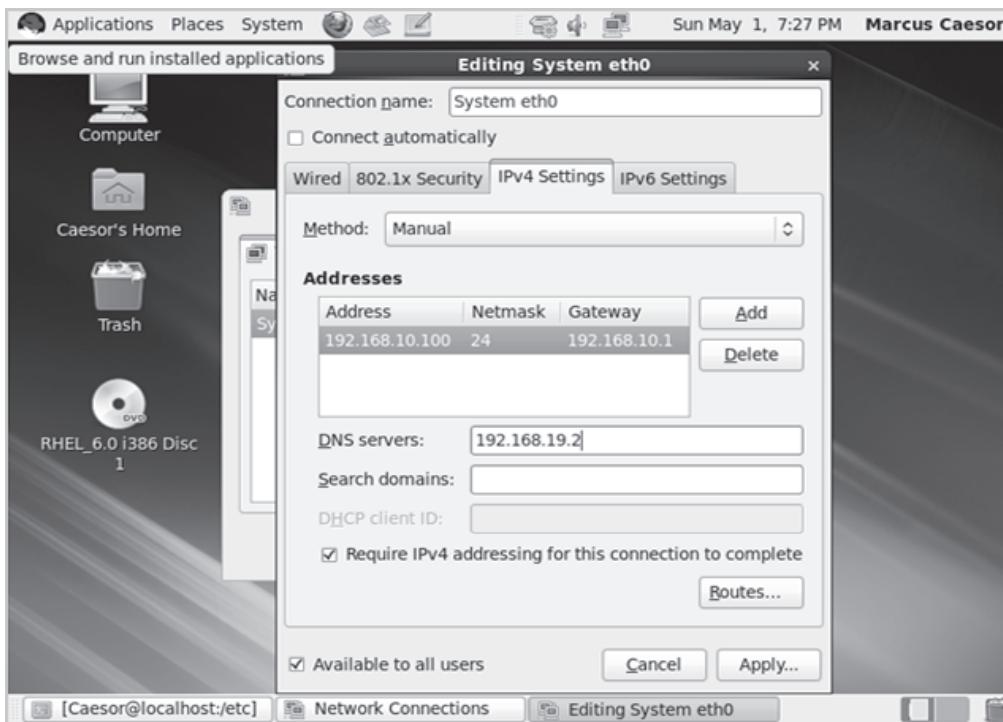


Figure 14.23: DNS Servers IP Address

Step 8 - Click Apply.

Step 9 - Enter the password and click Authenticate.

Step 10 - Click Close on the Network Connections dialog box.

Exercise - 6

Connecting to a remote computer

Step 1 - Log on to the RHEL system with the login credentials.

Step 2 - Click Applications → System Tools → Terminal to open the terminal.

Step 3 - At the command prompt, execute the following command to navigate to the Documents directory:

```
ssh Caesor@113.193.157.55
```

Note - Users should replace the IP address with a different IP address of the destination server.

Step 4 - Type the password to connect to the remote computer.

After the authentication is successful, users are connected to the remote system as shown in Figure 14.24.

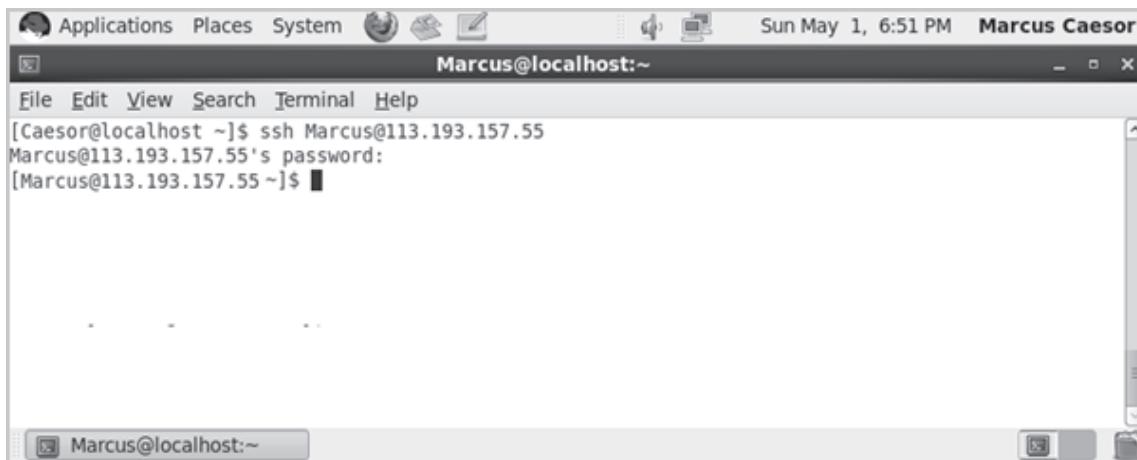


Figure 14.24: Remote System Connection

Step 5 - Type exit and press ENTER to close the connection to the remote computer.

The remote connection is closed as shown in Figure 14.25.

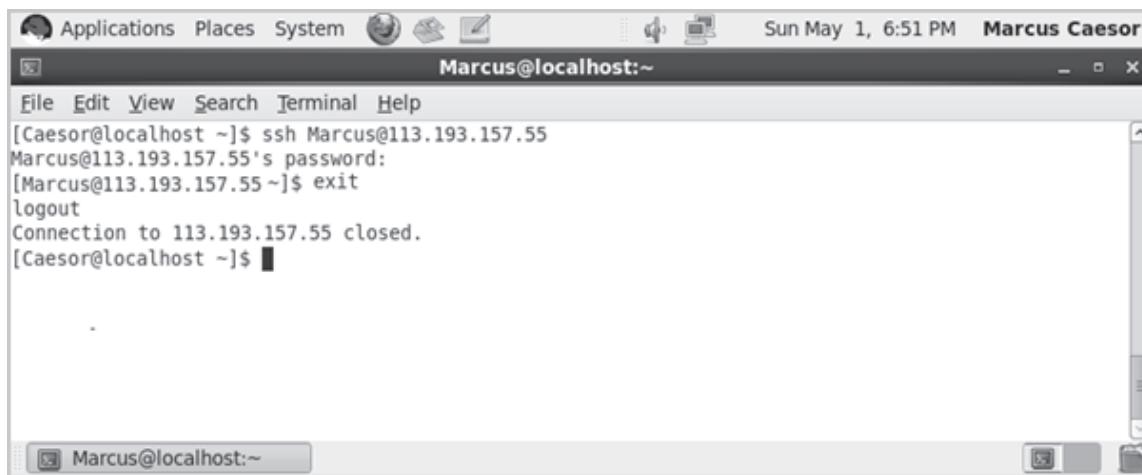


Figure 14.25: Closed Remote Connection

Step 6 - Type exit to close the terminal.

TechnoWise



Are you a
TECHIE GEEK
looking for updates?

Logon to

www.onlinevarsity.com

15.1 User Accounts

A user account provides a user the capability to access the computer with a specific set of privileges. Each user account is assigned a user name that is unique in a specific network. A user account is primarily an account that is created for a specific user or an application. Each user account is identified by a name that can either correspond to a user or an application. There are certain attributes that are identified for a user account when it is created. Some of these attributes are as follows:

- ➔ Name
- ➔ Group
- ➔ Home Directory
- ➔ UID

User accounts and UIDs are stored in the /etc/passwd file in the operating system. Each user, when created, is assigned a home directory. A user is also assigned a program, which is a shell that they can run while logging on.

Note - A UID is unique for each user account. The same UID is not assigned to a user account even when it is recreated with the same name. A new unique UID is assigned.

Some of the permissions that can be configured in a user account are as follows:

- ➔ To personalize desktops
- ➔ To restrict access to the user's files and folders to other users
- ➔ To share files with other users over the network
- ➔ To track certain set of activities for a specific user
- ➔ To gain full control over a home directory and its files and folders for a user

Multiple user accounts can be created on a computer. All these accounts are considered as local computer accounts. Each local user account can be used to log on to the computer. In the similar way, the network-based computer accounts can be added to a local computer to allow them to log on to the computer.

Each user that belongs to a specific computer has a dedicated folder that is called home folder. The settings that are configured by a user do not affect the settings of other users. These settings are maintained in the user profile. After logging on to a computer, a user is able to access the files and folders that reside in the home folder. A user has certain privileges to run applications. However, when the user logs off, the applications and other open documents automatically close.

When a user account is created, it should be protected with a password. While a user attempts to log on

to a RHEL computer, the password is required to perform a successful log on.

Note - It is recommended to assign a password to each user account that is created. Passwords prevent any unauthorized personnel to access a computer. Typically, the users tend to keep the password simple. However, it is recommended that the password should be complex. The password should contain alphanumeric and special characters.

15.1.1 Multiple User Accounts

In an enterprise scenario, there are thousands of users who access and share files over the network. The sharing of files over the network is typically restricted by the access given to specific users who have to access the files. Therefore, it is necessary for each user to have a specific user account. This helps the user to perform certain set of tasks on the local computer and on the network as well.

Consider a scenario where there are multiple users but there is only one user account named User1. Each user logs onto the networked computer using the same user account. In this scenario, if the network administrator has to assign permissions to a user, the administrator is unable to do so. A simple reason is because this user and the other users over the network are using the same user account. Also, the user is unable to share any confidential information with another user. This is one primary reason why there should be multiple user accounts and why each user should have their own specific user account.

When there are multiple user accounts, each user has their own specific set of privileges that can be similar or can differ in certain context. Users with higher privileges have more user rights over the files and directories stored in a computer. A user cannot read, write or execute the files of another user unless superuser rights are provided. To get read, write or execute rights for the files of another user, a user requires specific permissions. The permissions are granted either by the owner of the files or the superuser.

When a user installs RHEL operating system, few standard users are created. These standard users are configured in the /etc/passwd file.

Table 15.1 lists some of the standard users that are created during installation.

User	UID	Home Directory
root	0	/root
bin	1	/bin
daemon	2	/sbin
adm	3	/var/adm
mail	8	/var/spool/mail

Table 15.1: User Information

15.1.2 Group Accounts

A group is a collection of user accounts. Typically, a user account is used for assigning permissions to a specific folder on the network. Consider a scenario, where all users from the Finance department requires access to the Company_Finance folder on a server hosted on a network. All users require the same permissions. In this scenario, there are two methods that an administrator can use to assign permissions on the Company_Finance folder. The first method is to individually select each user account and assign permissions. However, this method is time consuming specifically if there are a large number of users. The second method is to create a group and add all the users from the Finance department in this group. When the users in the Finance department require permission to the folder, the administrator can simply add this group to the folder and define relevant permissions. The required permissions are applied to all the users in the group.

The factors to be considered to create a group and combine users are as follows:

- Role
- Departments
- Geographies
- Responsibilities

Note - Similar to the unique UID for each user account, each group is also assigned a unique GID.

Along with the standard users, there are a number of standard groups that are created.

Table 15.2 lists some of the standard groups that are created during installation.

Group	GID	Members
root	0	root
bin	1	root, bin, daemon
daemon	2	root, bin, daemon
sys	3	root, bin, adm
adm	4	root, adm, daemon

Table 15.2: Group Information

The ***id <user account>*** command syntax is used to display UID and GID of specific user. Only ***id*** command displays UID and GID of logged on user account. Figure 15.1 shows UID, GID, and groups of the root user.

```
login: root
Password:
Last login: Thu Apr  7 05:03:25 on tty1
[root@localhost ~]# id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10
(wheel) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@localhost ~]# _
```

Figure 15.1: UID, GID, and Groups of Root User

After a user is created, the user is given an own private group. However, this is not a limitation. Users can be added to more than one group. Adding users to the relevant group helps in assigning access to a large number of users with ease. It is important to note that if a file belongs to a group, all users in that group can share the file with the other users.

Changing Identities

Identities can also be changed while working as a user. A user can execute a command or perform a task using someone else's user account. The user can also use another user account that has higher privileges. Consider a scenario in which a user named xyz has to perform a task that requires superuser authority. The user xyz does not have superuser permissions to create a user account. In this scenario, the user xyz can change identity as a root user and create the user account.

Execute the following command:

```
su - root
```

When the command is executed, it prompts for the password. After the authentication is successful, an administrator is able to perform the tasks that require superuser privileges. After performing the task, press **ENTER** key at the command prompt to exit from the superuser account. It is important to remember that after a user gains the privileges of the root user, the user can again run the **su** command. Also, because the user has root privileges, the user is not prompted for the password while changing to any other user.

In Figure 15.2, the root user is using **su** command to switch to another user xyz. Note, the root user is not prompted for any password.

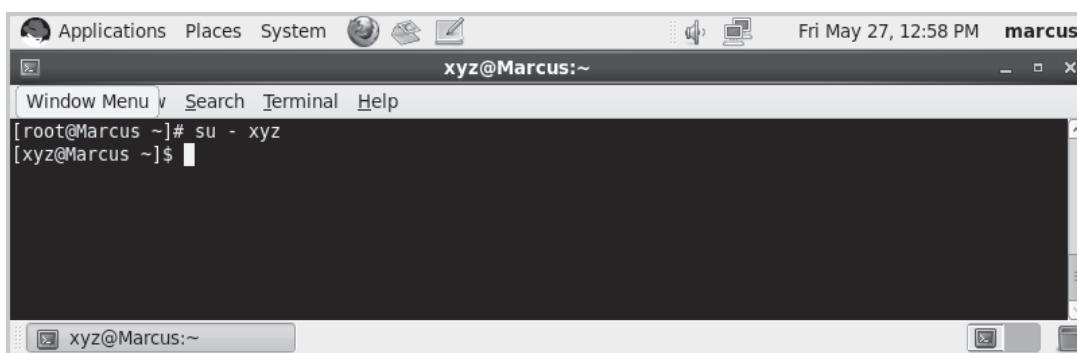


Figure 15.2: Root User Switching to User xyz Using su Command

The `exit` command is used to switch to the previous user account.

In Figure 15.3, the user root switched to user account xyz using the `su -xyz` command and switched back to the previous user account (root) using the `exit` command.

```
[root@Marcus ~]# su - xyz
[xyz@Marcus ~]$ exit
logout
[root@Marcus ~]#
```

Figure 15.3: Using `exit` Command to Switch to the Previous User Account

A user can assume to be any other user by using the `su` command with the assumed user account name.

```
# su - xyz
```

Assuming that a user named xyz has changed identities with another user named abc after executing this command. Now, xyz is able to perform the tasks that only abc had the privileges to do.

The `su` command is typically used with a - (hyphen). When this command is used with a - (hyphen), it has the following two effects:

Along with the user, it also switches to the user's home directory.

It uses the environmental variables that were being applied to the changed user.

After the execution of `su` command with a - (hyphen), the environment is changed according to the user that is specified. For example, when the `su - root` is executed, the current environment is set exactly as when a root user was logged on.

Note - A user should be cautious while using the root user with the `su` command. With the root user, the user gains absolute administrative rights to the computer where this command is executed.

Other than the `su` command, the `sudo` command can also be used. The `sudo` command runs the commands as a root user. The `sudo` command is often mistaken with the `su` command but there are noticeable differences between the two commands. However, the `sudo` command has a few distinct capabilities while the `su` command does not.

Some of the capabilities of `sudo` command are as follows:

- It can include another command.

- ➔ It uses definable constraints.
- ➔ It tracks the usage of all commands in a log file.
- ➔ It does not require the root password.
- ➔ It elevates the privileges of the user account being used with the command, such as root.

15.2 Privileges

A privilege is an attribute that is assigned to a user to execute specific tasks and functions. It is an ability that allows a user, an application or a process to override certain security constraints that otherwise are applied. A privilege helps the user to bypass certain restrictions and limitations that otherwise would have stopped the user from executing certain tasks on a system. A privilege can also be assigned to an application or process to gain certain capabilities in a system.

Consider a scenario in which a user does not have the privilege to add or delete another user. However, if this privilege is assigned to another user, xyz, the user is able to add or delete another user. With this privilege, xyz has higher privileges than any other user who does not have this privilege. In context to an application, it can be assigned higher privileges to override certain security constraints.

15.2.1 Root

In Windows operating system, an administrator is the superuser. The superuser has complete control over the operating system. Similarly, in the RHEL operating system, root is the superuser who has complete control over the operating system.

There are a number of key tasks that a root user can perform, for which the superuser rights are required. The key tasks are mostly system-related tasks that cannot be performed by a normal user.

Some of these tasks are as follows:

- ➔ Installing applications, such as third-party applications
- ➔ Installing and configuring devices, such as video adapter drivers
- ➔ Installing and configuring system services, such as Web server or FTP server
- ➔ Adding new users or deleting existing users from the RHEL operating system

The root user account virtually has the power to perform all operations and system level tasks in the operating system, which is not limited to the mentioned tasks.

The root account is listed in the /etc/passwd file. It has the UID marked as 0. If there is any user with UID as 0, it has the same set of privileges as the root user. The authentication of a root account is always done

using the local security files.

There are users who would require using the root user account to perform certain system-level operations. However, it is not advisable to grant everyone the rights of root user account. Instead, each user should log on using their respective user account and then use the `su` command to impersonate root user account. In this way, a record is maintained in the `/var/adm/syslog` for all the activities performed by the users.

Note - It is advisable to assign root access only to a few users. There are two specific reasons for which the root access should be limited only to a few users. The first reason is that more users have complete control of the operating system. The second reason is that as more users have root access, it makes accountability more difficult.

15.2.2 User

A root user has the power to perform virtually any operation in the RHEL operating system. On the other hand, a user, who is considered to be a normal user, has only restricted capabilities. The capabilities of a user also depend on the group of which they are a part of. As a user has restricted privileges, it is safer to use a user account than a root account, which can be a cause of concern if the users misuse this account.

Table 15.3 lists some of the basic differences between a root and a user.

Properties	Root User	Normal User
Security	Low	High
Privileges	Complete operating system	Own home directory
The <code>su</code> command to access another user	Password not required	Password required
UID	0	Dynamically assigned
GID	0	Dynamically assigned
Private group	Root	Username group
Console access	Unlimited	Limited

Table 15.3: Root and Normal User Differences

As mentioned in Table 15.3, most of the attributes of the root user account is predefined. This is not in the case of a user that is created by root or another user with superuser privileges.

Figure 15.4 highlights some of the differences that are mentioned in Table 15.3.

```
[root@localhost ~]# id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10
(wheel) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@localhost ~]# _____ Root
xyz@Marcus ~]$ id
id=501(xyz) gid=501(xyz) groups=501(xyz) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0
c1023
xyz@Marcus ~]$ [ ] xyz
```

Figure 15.4: Root and Normal User's UID and GID

15.2.3 User Manager

User Manager is used in creating a user account in the RHEL graphical environment. The User Manager displays the existing user accounts and the details in RHEL as shown in Figure 15.5.

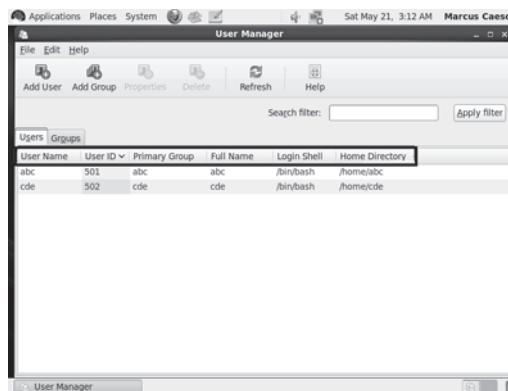


Figure 15.5: User Manager Details

With the User Manager, users can add, modify or delete users. Similarly, users can also add, modify or delete groups. In addition, users can be filtered using the built-in search function.

Note - The superuser, which is typically root, password is required to start the User Manager.

15.3 useradd Command

The second method is using the `useradd` command from the command prompt. To add a user using the `useradd` command, perform the following steps:

At the command prompt, type the `useradd <username>` command to create a user account, the `useradd` command can use `username` as the input parameter, which is the basic requirement to create a user account. Along with this parameter, more parameters can be used, such as the parameters defined in Table 15.4. A user with the specified name is created. However, it is important to note that the user account is created in the locked mode. In locked mode, the user account is currently not in a usable state until the password for the user account is defined.

To define the password, use the following syntax:

```
passwd <password>
```

After the password is set, the user account is released from the locked state and is enabled.

With the `useradd` command, a number of parameters can be set. Table 15.4 lists the parameters of the `useradd` command.

Parameter	Description
<code>-c '<comment>'</code>	Adds a comment for the user account.
<code>-d <home-dir></code>	Defines any other home directory. The default home directory is <code>/home/<username></code> .
<code>-e <date></code>	Sets the expiry date of the account. The format is <code>YYYY-MM-DD</code> .
<code>-f <days></code>	Sets the number of days for which the user account is kept active after the password expires. The value of '0' disables the account immediately after the password expires. The value of '-1' keeps the account active.
<code>-g <group-name></code>	Adds the default group name or number to a user.
<code>-G <group-list></code>	Adds any additional group name or number other than the default group to a user.
<code>-m</code>	Creates the home directory if it does not already exist.
<code>-M</code>	Skips the home directory creation.
<code>-N</code>	Skips the creation of a user's private group.
<code>-p <password></code>	Encrypts the password with crypt.
<code>-r</code>	Creates a system account that has a UID less than 500. The home directory is not created with this parameter.
<code>-s</code>	Defines the default login shell.
<code>-u <uid></code>	Defines the UID for the user, which has to be greater than 500.

Table 15.4: `useradd` Command Parameters

15.3.1 Files and Directories

There are a number of operations that can be performed at the file system level. Some of these operations are considered as basic operations, such as copying and deleting files.

Important Directories

File System Hierarchy Standard defines the structure and naming convention for directories that should exist within the operating systems that are similar to UNIX. There are a set of directories and subdirectories that are used for specific purposes.

Table 15.5 lists the directories and their purposes.

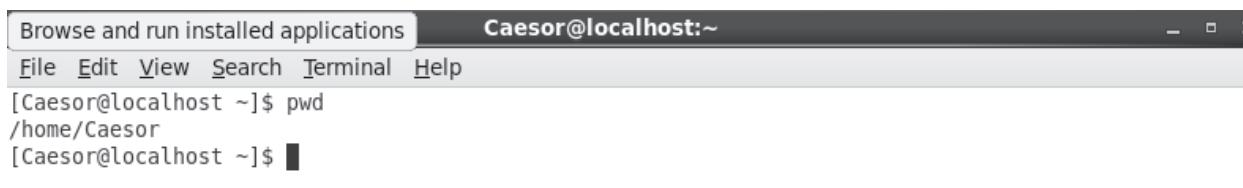
Directory Name	Function
/bin/	Contains essential commands that are used by administrators and users.
/usr/bin/	Contains common commands that are used by administrators and users.
/sbin/	Contains essential commands that are used by administrators.
/usr/sbin/	Contains common commands that are used by administrators.
/tmp/	Contains the temporary files for all users.
/usr/local/	Contains locally installed applications.
/usr/share/man/	Contains manual pages.
/usr/src/	Contains source code.
/var/	Contains variable files. Spool and log files are stored here.
/var/log/	Contains log files.
/etc/	Contains configuration files.
/proc/	Contains Kernel virtual file system.
/dev/	Contains device files.

Table 15.5: FHS Directories

A normal user would have restriction on some of the key directories. A root user would have permission on all the directories.

Current Working Directory

Consider a scenario where a user is working on the command line and has to know the absolute path of the current location. To know the absolute path, the user can run the `pwd` command. In Linux or UNIX, each shell or a system process is assigned a Current Working Directory (CWD). To know the cwd of a process, the user must ENTER `pwd` at the command prompt. Figure 15.6 displays the output of the `pwd` command.



The screenshot shows a terminal window with the following content:

```
Browse and run installed applications Caesor@localhost:~ - □ >
File Edit View Search Terminal Help
[Caesor@localhost ~]$ pwd
/home/Caesor
[Caesor@localhost ~]$ █
```

Figure 15.6: Output of `pwd` Command

File and Directory Names

RHEL and all Linux and UNIX operating systems are case sensitive to file and directory names. A file name

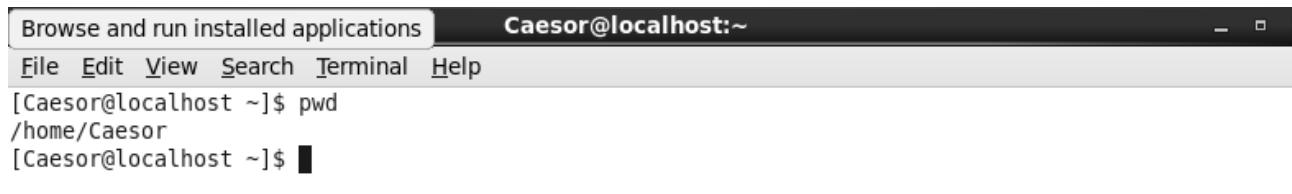
can use all types of characters except the forward slash (/). A file or directory name can extend up to 255 characters.

Absolute and Relative Pathnames

To understand the concept of absolute and relative pathnames, consider this scenario. While working on the command line, users frequently change to a different directory than the current one. A user can change the directory either by using the absolute path or the relative path.

The difference between the absolute and relative path names are as follows:

- **Absolute** - It is also known as the full path. When a user is changing to another directory using `cd <directory>` command, the user can define the absolute path in place of directory as the parameter. The absolute path starts with the root directory and then defines the complete path. An example of absolute path is `/home/abc/Documents`. The first slash (/) denotes the root directory and the remaining path is the directory structure. Figure 15.7 displays the absolute path.

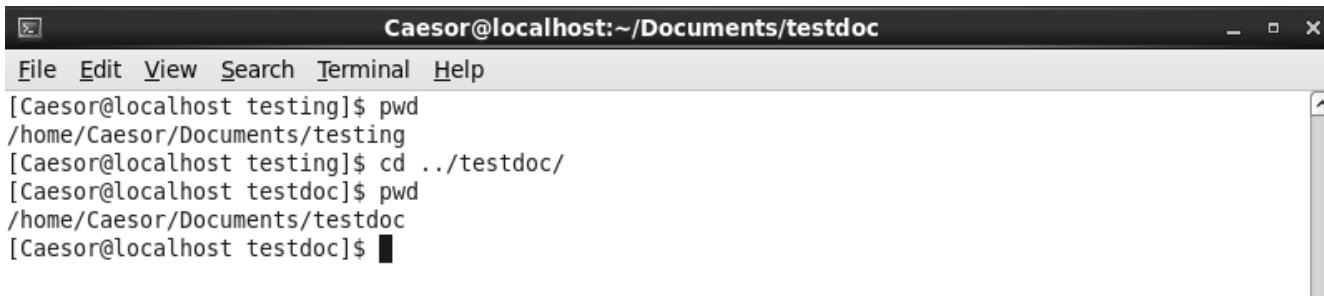


A screenshot of a terminal window titled "Caesor@localhost:~". The window has a standard OS X-style title bar with icons for close, minimize, and maximize. The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The main pane shows the command line history:

```
[Caesor@localhost ~]$ pwd
/home/Caesor
[Caesor@localhost ~]$
```

Figure 15.7: Absolute Path

- **Relative** - Relative path does not require a complete path. A user can specify one or more directories with two dots (..). For example, if the user is in the `/home/xyz/Documents/testing` directory and has to go to the `/home/xyz/Documents/testdoc` directory, the user can run the command `cd ../testdoc`. Without going to a previous directory, the user can easily navigate to another subdirectory within the same parent directory. Figure 15.8 displays the relative path.



A screenshot of a terminal window titled "Caesor@localhost:~/Documents/testdoc". The window has a standard OS X-style title bar with icons for close, minimize, and maximize. The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The main pane shows the command line history:

```
[Caesor@localhost testing]$ pwd
/home/Caesor/Documents/testing
[Caesor@localhost testing]$ cd ../testdoc/
[Caesor@localhost testdoc]$ pwd
/home/Caesor/Documents/testdoc
[Caesor@localhost testdoc]$
```

Figure 15.8: Relative Path

Changing Directories

When a user has to change directories to accomplish a task at the command prompt, the user has to first execute the following command:

`cd <directory>`

Execute the following command to change directory:

```
#cd -L -P <directory>
```

where,

The -L parameter forces the symbolic links to be followed.

The -P parameter uses physical directory structure.

Figure 15.9 displays the use of the cd command.

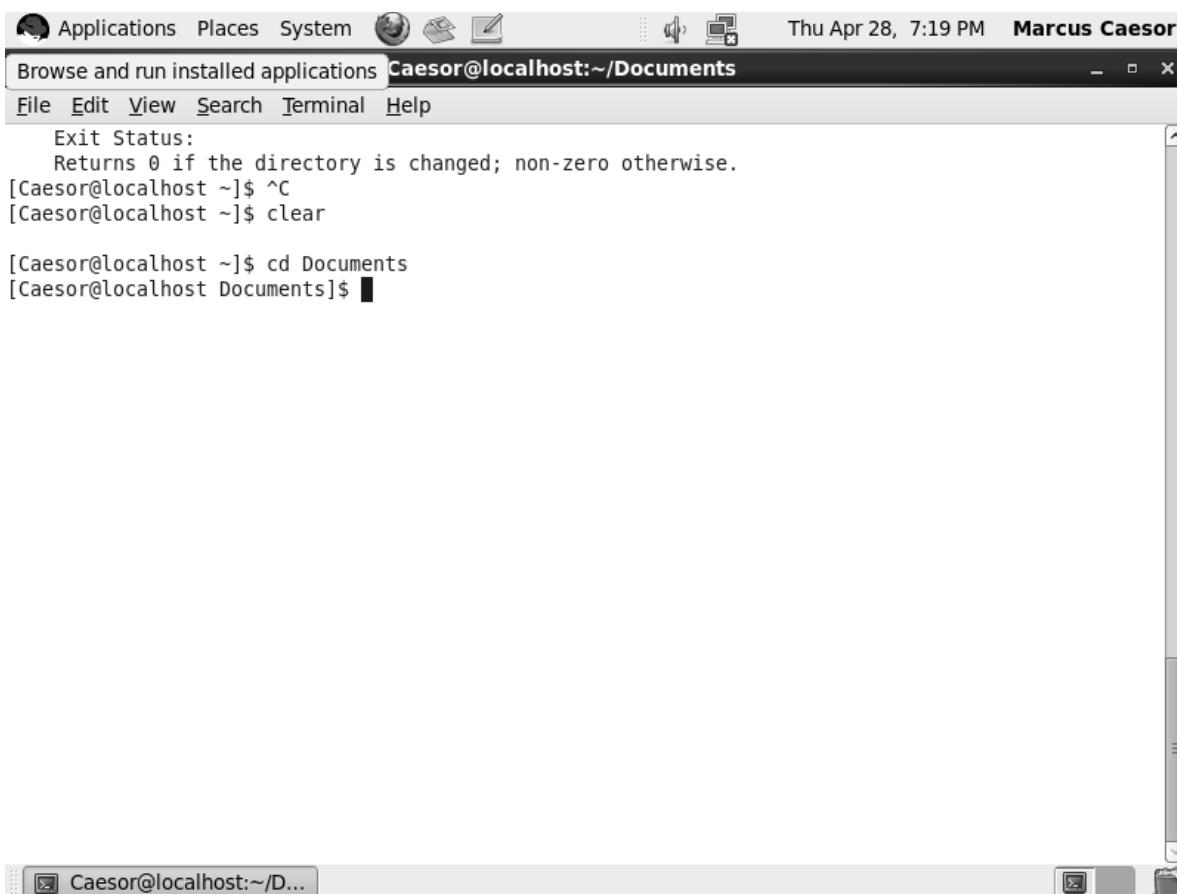


Figure 15.9: Using the cd Command

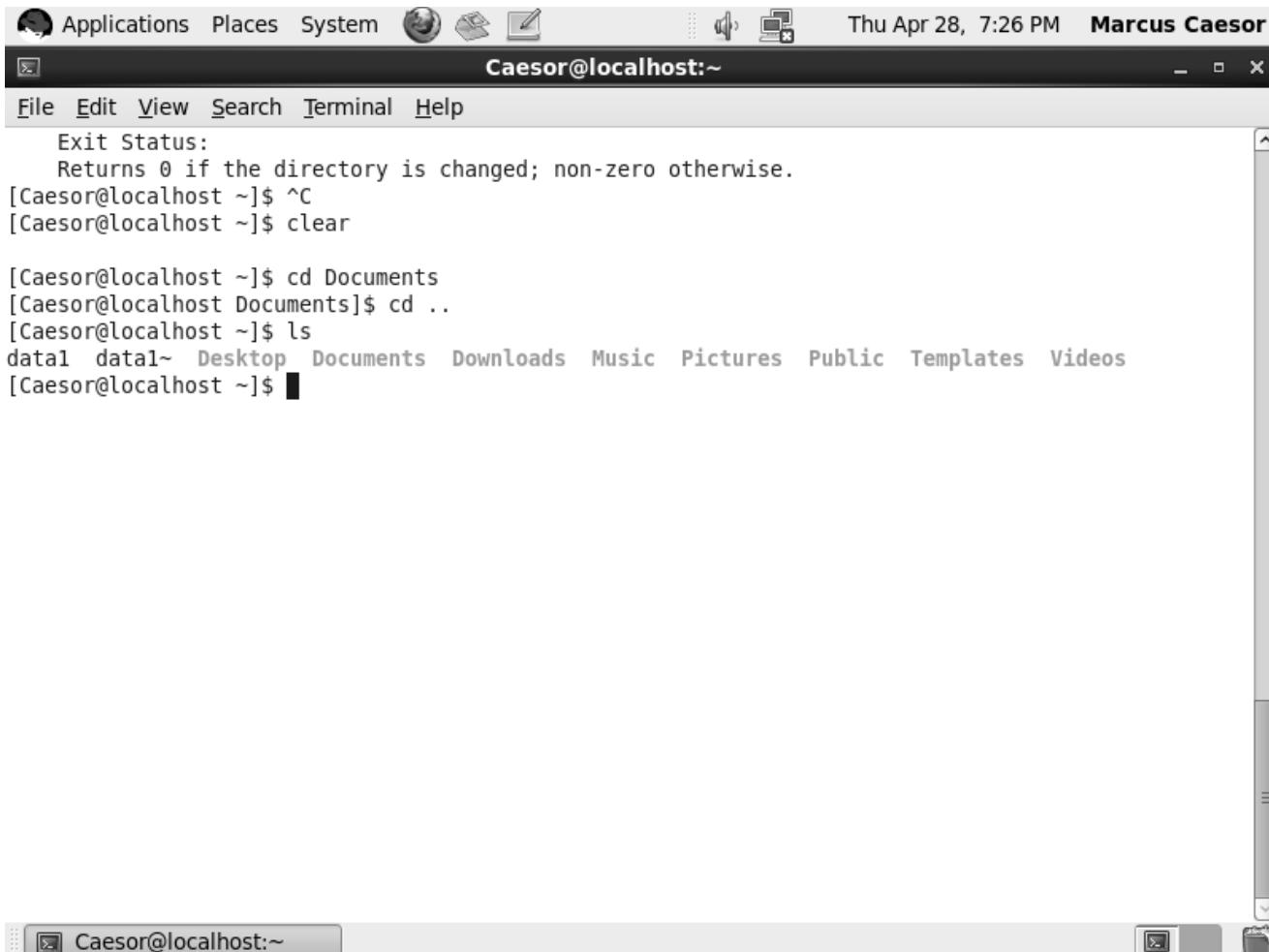
It is important to note that the shell prompt also changes when a user changes the directory. The directory name is also added to the shell prompt. In Figure 15.10, the directory is changed to /Documents in user's home directory.

Following is the shell prompt that is shown after changing the directory:

```
[Caesor@localhost Documents]
```

The cd command also allows a user to navigate between directories using the absolute or relative path.

Figure 15.10 displays the example of the cd command.



The screenshot shows a terminal window titled "Caesor@localhost:~". The window includes a menu bar with File, Edit, View, Search, Terminal, and Help. The terminal itself displays the following command-line session:

```
Exit Status:  
Returns 0 if the directory is changed; non-zero otherwise.  
[Caesor@localhost ~]$ ^C  
[Caesor@localhost ~]$ clear  
  
[Caesor@localhost ~]$ cd Documents  
[Caesor@localhost Documents]$ cd ..  
[Caesor@localhost ~]$ ls  
data1 data1~ Desktop Documents Downloads Music Pictures Public Templates Videos  
[Caesor@localhost ~]$ █
```

Figure 15.10: Example of cd Command

Note - The **cp** command does not preserve the Access Control List (ACL) when a file is moved. On the other hand, the **mv** command preserves the original ACL of the files that are being moved.

Copying Files and Directories

The **cp** command is used to copy files and directories from source to the destination location.

The syntax for the **cp** command is as follows:

```
# cp [options] file destination
```

If a user has to directly move to the home directory (Refer to Figure 15.11), the user should run the following command:

```
$ cd
```

The screenshot shows a desktop environment with a terminal window open. The window title is "Access documents, folders and network places". The terminal session shows the following command history and output:

```
[Caesor@localhost ~]$ cd Do  
Documents/ Downloads/  
[Caesor@localhost ~]$ cd Documents/  
[Caesor@localhost Documents]$ ls  
file1.txt file2.txt file3.txt file4.txt file5.txt testdoc testing  
[Caesor@localhost Documents]$ cd testdoc  
[Caesor@localhost testdoc]$ ls  
[Caesor@localhost testdoc]$ cd  
[Caesor@localhost ~]$ ls  
data1 data1~ Desktop Documents Downloads Music Pictures Public Templates Videos  
[Caesor@localhost ~]$ █
```

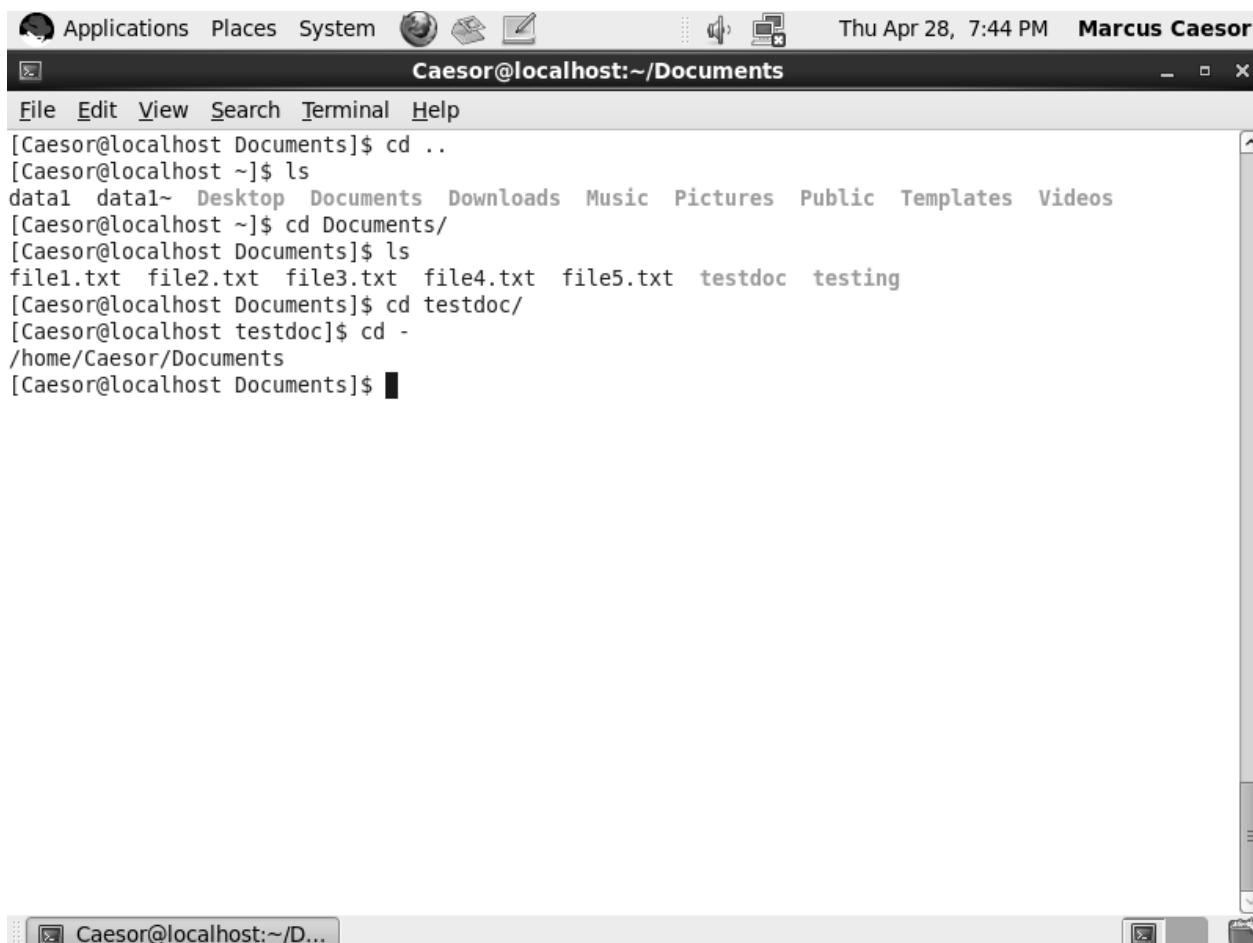
The terminal window is part of a desktop interface, with a menu bar at the top and a taskbar at the bottom.

Figure 15.11: Moving to the Home Directory

If the user has to move to the previous directory, the user should run the following command:

```
# cd -
```

Figure 15.12 displays the output of this command.



The screenshot shows a terminal window with a dark grey header bar. The header bar contains icons for Applications, Places, System, and several system status indicators. The date and time 'Thu Apr 28, 7:44 PM' and the user 'Marcus Caesor' are also visible. Below the header is a menu bar with options File, Edit, View, Search, Terminal, and Help. The main window area displays a command-line session:

```
[Caesor@localhost Documents]$ cd ..
[Caesor@localhost ~]$ ls
data1 data1~ Desktop Documents Downloads Music Pictures Public Templates Videos
[Caesor@localhost ~]$ cd Documents/
[Caesor@localhost Documents]$ ls
file1.txt file2.txt file3.txt file4.txt file5.txt testdoc testing
[Caesor@localhost Documents]$ cd testdoc/
[Caesor@localhost testdoc]$ cd -
/home/Caesor/Documents
[Caesor@localhost Documents]$
```

The terminal window has scroll bars on the right and bottom. The title bar of the window is 'Caesor@localhost:~/Documents'. The bottom of the window shows a toolbar with icons for minimize, maximize, and close.

Figure 15.12: Output of cd Command

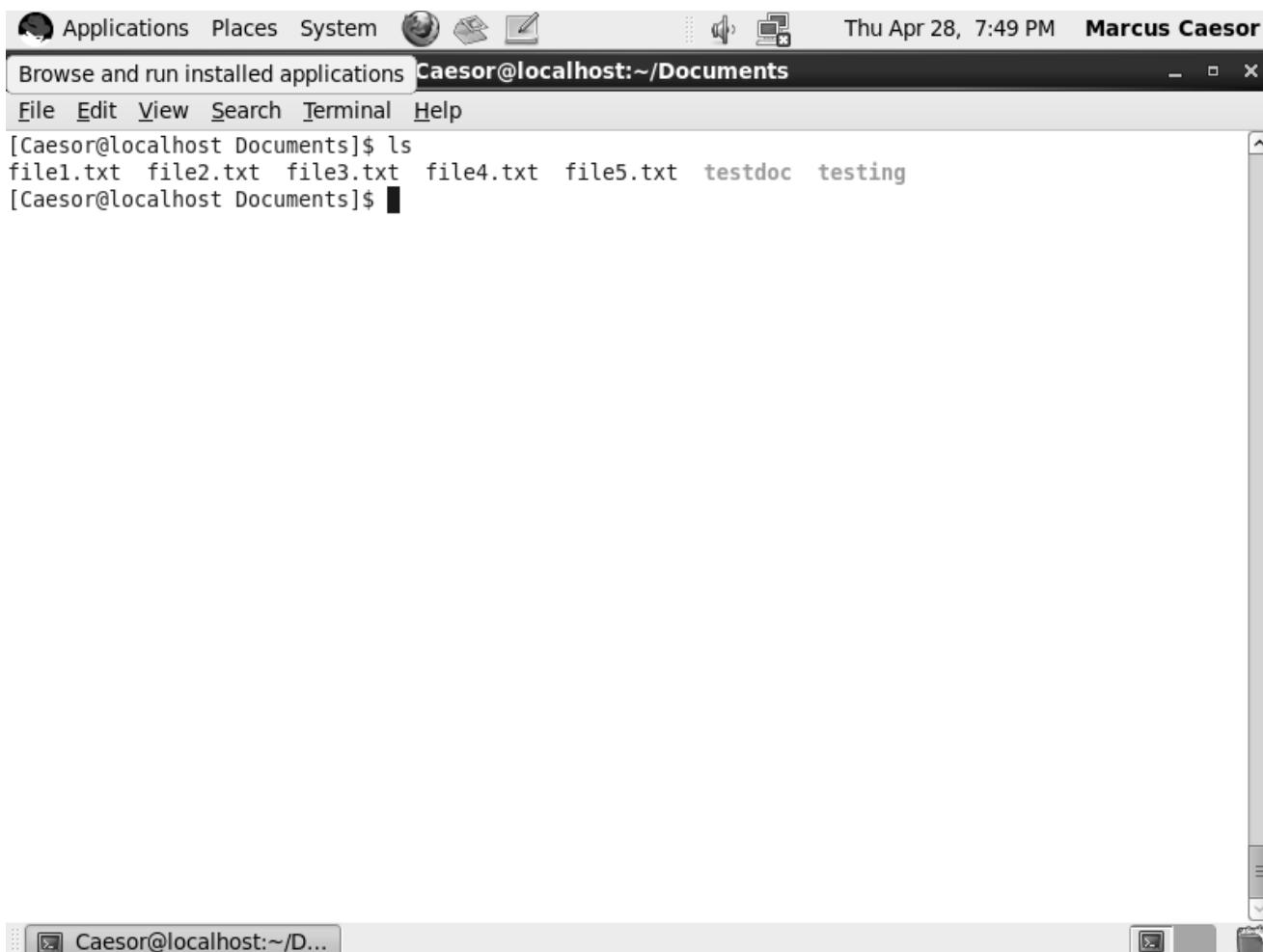
Listing Directory Contents

In some cases, a user has to list down the files in a directory. Consider a scenario where the user is searching for a particular file in a directory that has hundreds of files. The user is not sure if the file exists in this directory. In this case, the user can list all files and then scans through them to locate the file.

The syntax for the **ls** command is as follows:

```
ls [options] [files_or_dirs]
```

To list files, the user should run the **ls** command at the command prompt. Figure 15.13 displays an example of **ls** command.



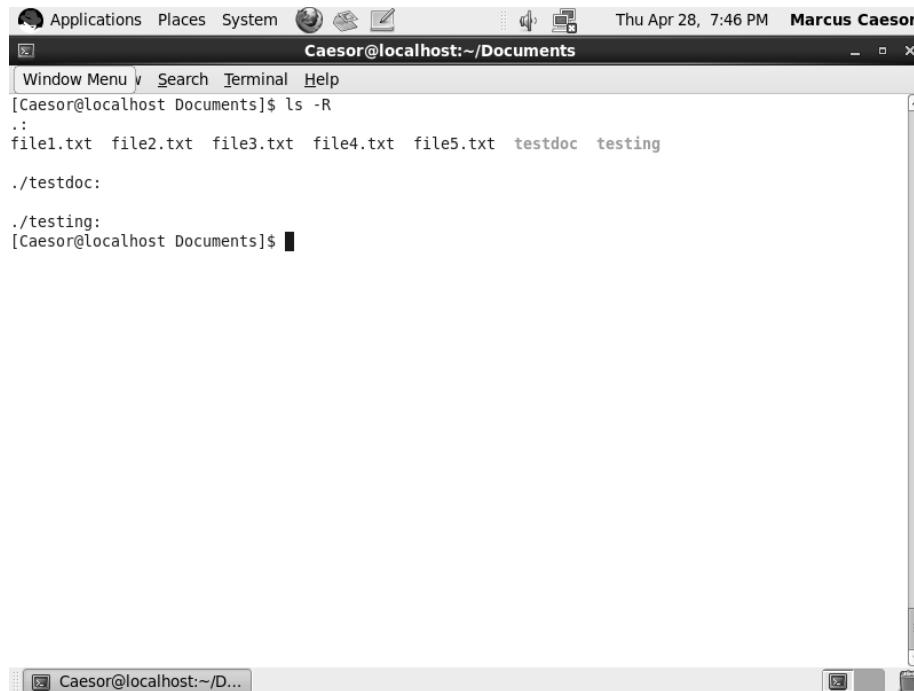
The screenshot shows a terminal window titled "Caesor@localhost:~/Documents". The window has a standard Linux desktop interface with a menu bar (Applications, Places, System) and a toolbar. The terminal itself displays the command [Caesor@localhost Documents]\$ ls followed by the output: file1.txt file2.txt file3.txt file4.txt file5.txt testdoc testing. The cursor is visible at the end of the command line.

Figure 15.13: Example of **ls** Command

Some of the key parameters that a user can use with the **ls** command are as follows:

- **ls -a** - Displays the hidden files in a directory
- **ls -l** - Displays extra information about the files
- **ls -R** - Recurses through directories to display files in the subdirectories

An example of the `ls -R` command is shown in Figure 15.14.



The screenshot shows a terminal window titled "Caesor@localhost:~/Documents". The window contains the following text:

```
Applications Places System Window Menu Search Terminal Help
[Caesor@localhost Documents]$ ls -R
.:
file1.txt file2.txt file3.txt file4.txt file5.txt testdoc testing
./testdoc:
./testing:
[Caesor@localhost Documents]$
```

Figure 15.14: Example of `ls -R` Command

15.3.2 Copying and Moving Files

Users are often required to copy or move files from one directory to other. The copy operation slightly differs from the move operation. In copy operation, the source file is retained as is and another copy of the source file is created at the defined destination.

In a move operation, the source file is deleted from the source location and moved to the destination location. Both are used in different situations. For example, if a user is backing up files from the data directory to another directory, the user copies files. However, assume that the user has to replace the hard disk on the computer. In this situation, the user would move the files from the source hard disk to the destination hard disk.

With the `mv` and `cp` command, a user can use the wildcards to filter out the files that user have to either copy or move. In this case, the user uses wildcards, such as `*` and `?` to filter the copy or move operations.

Note - The `cp` command does not preserve the Access Control List (ACL) when a file is moved. On the other hand, the `mv` command preserves the original ACL of the files that are being moved.

Copying Files and Directories

The `cp` command is used to copy files and directories from source to the destination location.

The syntax for the `cp` command is as follows:

```
# cp [options] file destination
```

Users can also copy more than one file to a destination directory by using the following command;

```
# cp [options] file1 file2 destination
```

Copying Files and Directories: The Destination

When copying files and directories, the rules to be observed are as follows:

- When copying the file to a directory, the file is copied to the destination directory.
- When copying the file and the destination directory does not exist, a file with the defined destination directory name is created.
- When copying the file and the destination is not a directory but a file, the destination file is overwritten.

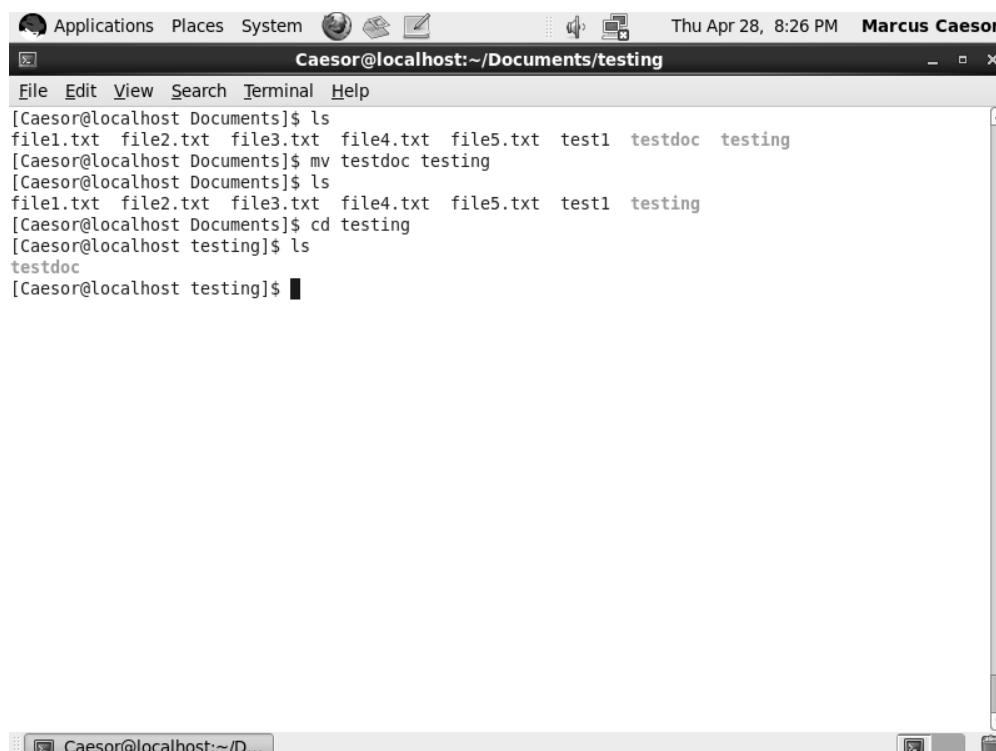
Moving and Renaming Files and Directories

If a user has to move files and directories from one location to another location, the user must execute the **mv** command to move files. The syntax for mv command is as follows:

```
mv [options] file destination
```

Similar to the **cp** command, the user can also move multiple files at a time, if the specified destination is a directory.

Figure 15.15 displays moving the **testdoc** directory to another directory named **testing**.



The screenshot shows a terminal window with the following session:

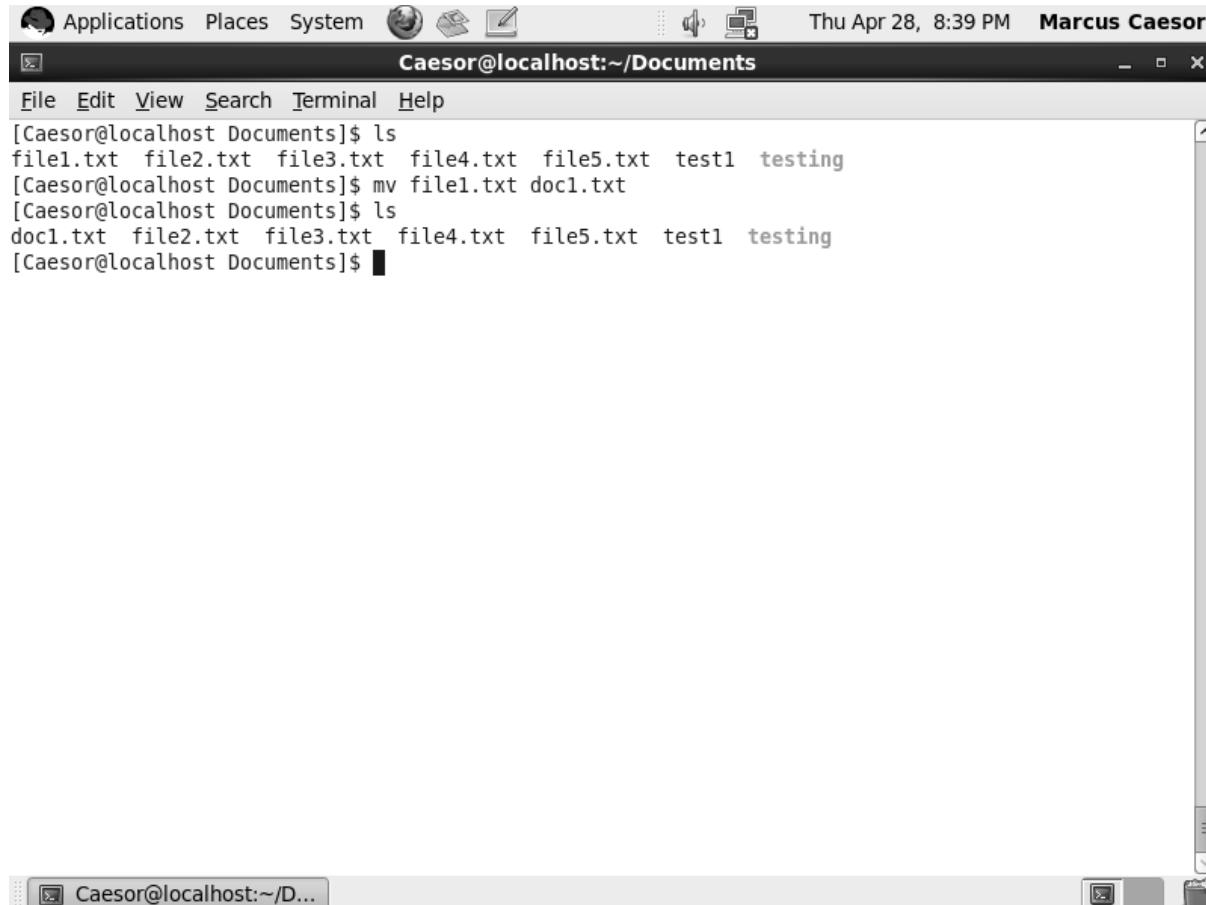
```
[Caesor@localhost Documents]$ ls
file1.txt file2.txt file3.txt file4.txt file5.txt test1 testdoc testing
[Caesor@localhost Documents]$ mv testdoc testing
[Caesor@localhost Documents]$ ls
file1.txt file2.txt file3.txt file4.txt file5.txt test1 testing
[Caesor@localhost Documents]$ cd testing
[Caesor@localhost testing]$ ls
testdoc
[Caesor@localhost testing]$
```

Figure 15.15: Moving the **testdoc** Directory

The `mv` command can also be used for renaming files. The following syntax should be used for renaming files:

```
mv filename new_filename
```

Figure 15.16 displays the output of the `mv` command to rename files.



The screenshot shows a terminal window titled "Caesor@localhost:~/Documents". The window has a standard Linux desktop interface with icons for Applications, Places, System, and others at the top. The title bar also shows the date and time: "Thu Apr 28, 8:39 PM" and the user name "Marcus Caesor". The terminal content is as follows:

```
[Caesor@localhost Documents]$ ls
file1.txt file2.txt file3.txt file4.txt file5.txt test1 testing
[Caesor@localhost Documents]$ mv file1.txt doc1.txt
[Caesor@localhost Documents]$ ls
doc1.txt file2.txt file3.txt file4.txt file5.txt test1 testing
[Caesor@localhost Documents]$
```

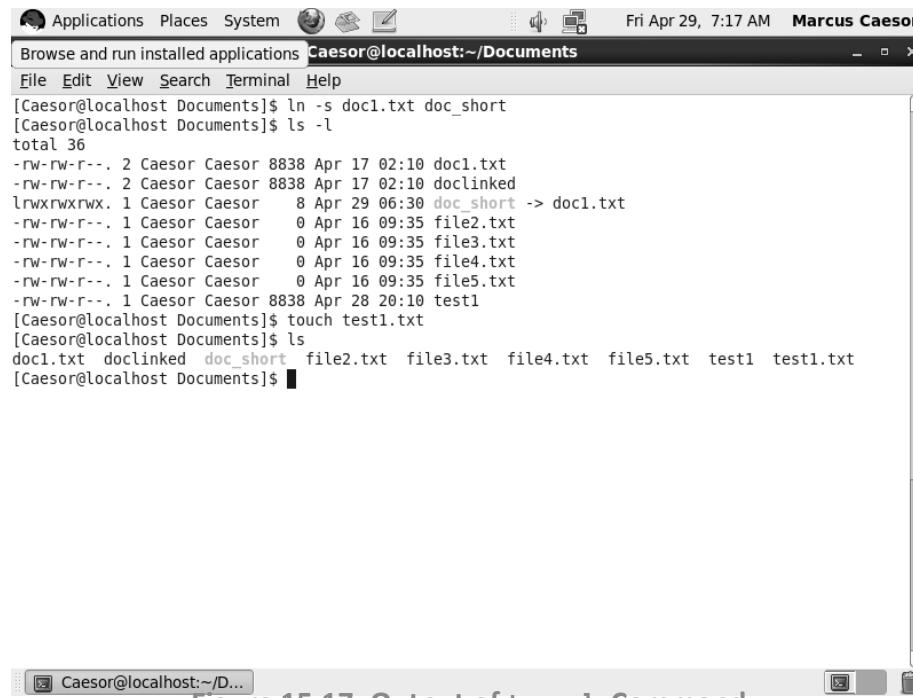
Figure 15.16: Output of the `mv` Command

Creating and Removing Files

A user can create or remove files depending on the requirements. To create a file, the user can use the `touch` command. The syntax for the `touch` command is as follows:

```
# touch <filename>
```

Figure 15.17 displays the output of the **touch** command.



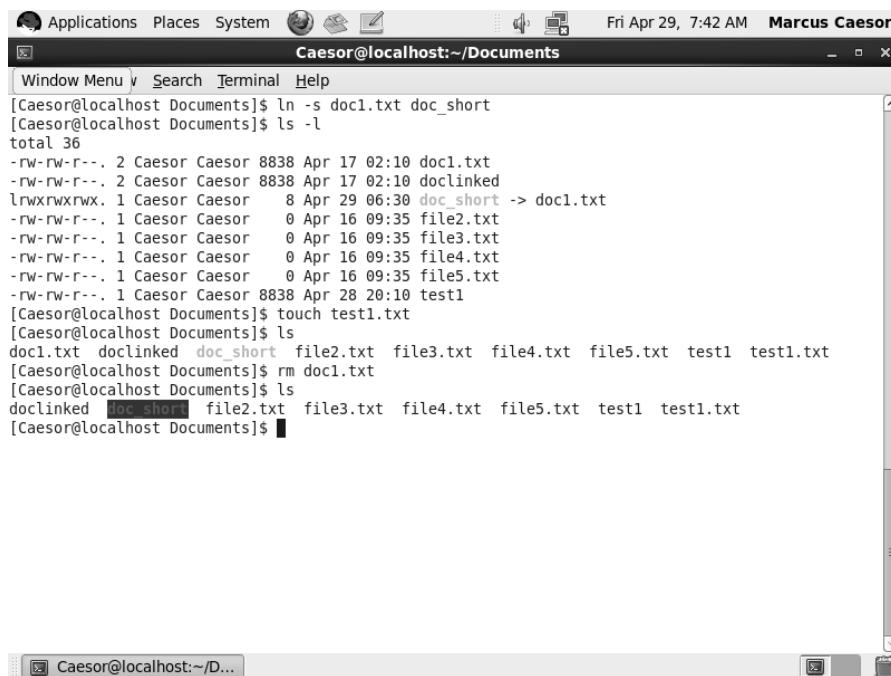
```
[Caesor@localhost Documents]$ ln -s doc1.txt doc_short
[Caesor@localhost Documents]$ ls -l
total 36
-rw-rw-r-- 2 Caesor Caesor 8838 Apr 17 02:10 doc1.txt
-rw-rw-r-- 2 Caesor Caesor 8838 Apr 17 02:10 doclinked
lrwxrwxrwx. 1 Caesor Caesor 8 Apr 29 06:30 doc_short -> doc1.txt
-rw-rw-r-- 1 Caesor Caesor 0 Apr 16 09:35 file2.txt
-rw-rw-r-- 1 Caesor Caesor 0 Apr 16 09:35 file3.txt
-rw-rw-r-- 1 Caesor Caesor 0 Apr 16 09:35 file4.txt
-rw-rw-r-- 1 Caesor Caesor 0 Apr 16 09:35 file5.txt
-rw-rw-r-- 1 Caesor Caesor 8838 Apr 28 20:10 test1
[Caesor@localhost Documents]$ touch test1.txt
[Caesor@localhost Documents]$ ls
doc1.txt doclinked doc_short file2.txt file3.txt file4.txt file5.txt test1 test1.txt
[Caesor@localhost Documents]$
```

Figure 15.17: Output of **touch** Command

To remove a file, users can use the **rm** command. The syntax for the **rm** command is as follows:

```
# rm <filename>
```

Figure 15.18 displays the output of the **rm** command.



```
[Caesor@localhost Documents]$ ln -s doc1.txt doc_short
[Caesor@localhost Documents]$ ls -l
total 36
-rw-rw-r-- 2 Caesor Caesor 8838 Apr 17 02:10 doc1.txt
-rw-rw-r-- 2 Caesor Caesor 8838 Apr 17 02:10 doclinked
lrwxrwxrwx. 1 Caesor Caesor 8 Apr 29 06:30 doc_short -> doc1.txt
-rw-rw-r-- 1 Caesor Caesor 0 Apr 16 09:35 file2.txt
-rw-rw-r-- 1 Caesor Caesor 0 Apr 16 09:35 file3.txt
-rw-rw-r-- 1 Caesor Caesor 0 Apr 16 09:35 file4.txt
-rw-rw-r-- 1 Caesor Caesor 0 Apr 16 09:35 file5.txt
-rw-rw-r-- 1 Caesor Caesor 8838 Apr 28 20:10 test1
[Caesor@localhost Documents]$ touch test1.txt
[Caesor@localhost Documents]$ ls
doc1.txt doclinked doc_short file2.txt file3.txt file4.txt file5.txt test1 test1.txt
[Caesor@localhost Documents]$ rm doc1.txt
[Caesor@localhost Documents]$ ls
doclinked doc_short file2.txt file3.txt file4.txt file5.txt test1 test1.txt
[Caesor@localhost Documents]$
```

Figure 15.18: Output of **rm** Command

Creating and Removing Directories

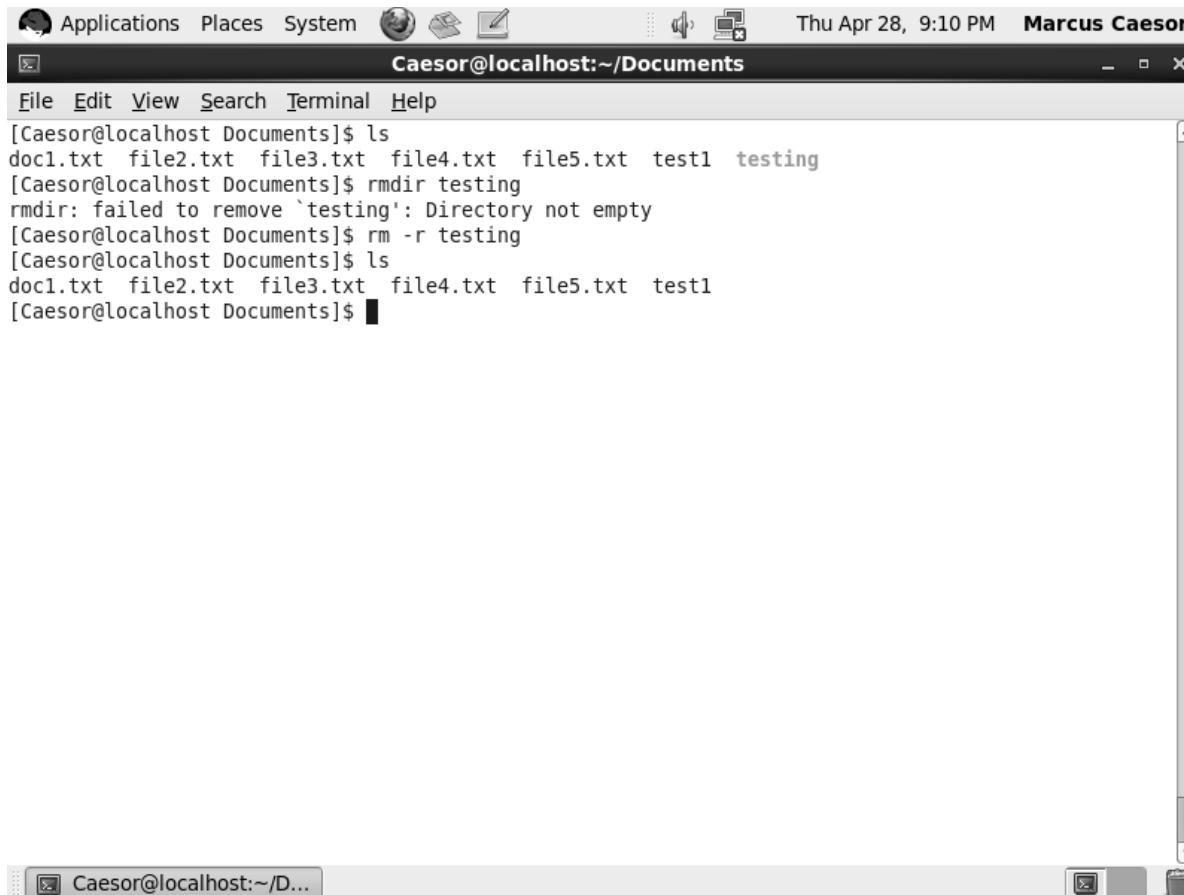
A user can also create and remove directories in RHEL. To create a directory, the user must use the `mkdir` command, which can take the absolute or relative path as input to create the directory. The `mkdir` command uses the following syntax:

```
mkdir [OPTION] directory
```

For example, if a user has to create a directory named `testing`, the user should execute the following command:

```
mkdir testing
```

There are two different commands to remove a directory. The `rmdir` removes empty directories, but not the directories with the files. If a directory, which contains files, has to be removed, the `rm` command with `-r` parameter must be used. An example of the `rm` command is shown in Figure 15.19.



The screenshot shows a terminal window titled "Caesor@localhost:~/Documents". The window includes a menu bar with File, Edit, View, Search, Terminal, and Help. The terminal content shows the following session:

```
[Caesor@localhost Documents]$ ls
doc1.txt file2.txt file3.txt file4.txt file5.txt test1 testing
[Caesor@localhost Documents]$ rmdir testing
rmdir: failed to remove 'testing': Directory not empty
[Caesor@localhost Documents]$ rm -r testing
[Caesor@localhost Documents]$ ls
doc1.txt file2.txt file3.txt file4.txt file5.txt test1
[Caesor@localhost Documents]$
```

Figure 15.19: Example of `rm` Command

15.4 Advanced Operations

Various advanced operations can be performed at the file system level. Some of the operations are automatically performed when a file or directory is created, moved or deleted. Some operations, such as creating links to files are performed by a user.

The advanced operations can also include transferring files from a local system to a remote system. However, depending on the data criticality, users can choose to transfer the files in a secure or non secure manner.

15.4.1 Inode and Dentries

A dentry is a short name for directory entry. Linux kernel uses dentry to keep track of the hierarchy of files in directories. Each dentry maps an inode number to a file name and a parent directory. In short, dentry is responsible for mapping the name identifiable by humans to the inode number, which is an index node that is identified by a number. An inode number is a unique number that is linked with a file name. The inode number information is maintained in the inode table, which contains the information of all files that exist on the ext2, ext3, or ext4 file system. The inode table maintains metadata information of each file.

The following details are included in the metadata:

- User ID (UID)
- Group ID (GID)
- Permissions assigned to the file
- File type
- File size
- Time stamps
- Pointers that point to the data blocks on the hard disk
- Location of the file

With different types of operations that are performed on the files, inodes are either created or modified. For example, when users copy a file from one directory to another directory, a new inode number is assigned in the inode table. A new dentry is also created and then linked with the inode table. If this association is not done, the file would not be locatable by the file system. The file is finally created with the required data from the source file.

If the file is moved from one directory to another directory on the same file system, the inode table is updated with minimal changes. Only the time stamp and data location is changed. The remaining information in inode is not changed. When a file is moved, a new dentry is created with a new file. As this is a move operation, the old dentry along with the old file name is deleted.

When removing files, the inode numbers are updated and the inode count is decremented with the number of files that are deleted. These inode numbers are freed up and made available for the new files.

The directory entries are also deleted. The data block held the file is freed up and put on the free list.

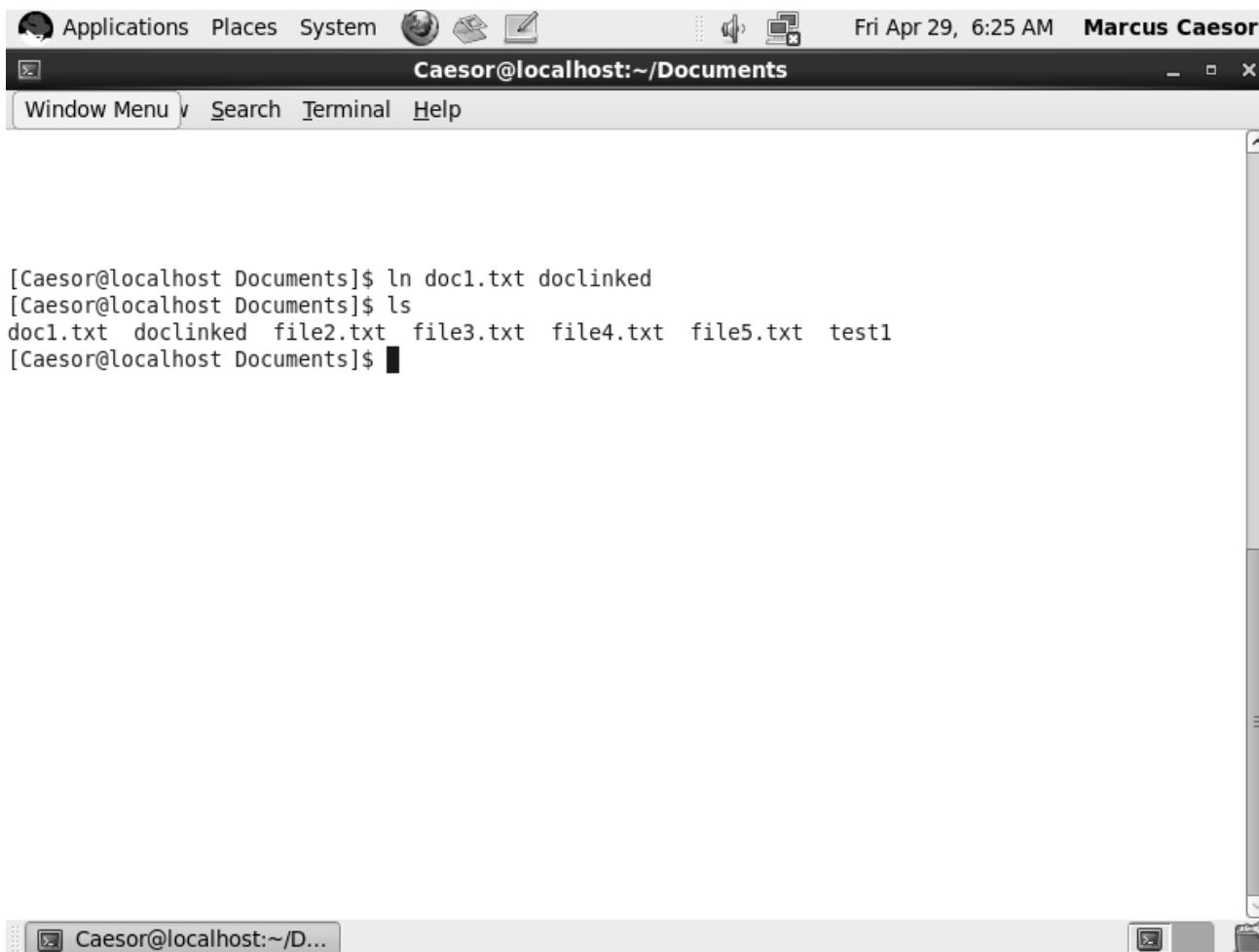
Create Symbolic Links and Hard Links

A hard link is a physical entry of a file. Each file that exists on the file system has one hard link assigned. If the hard link is deleted, the reference to the file is deleted and the file no longer exists. When a hard link is created, the link count is incremented and if the hard link is deleted, the link count is decremented. If the link count falls to zero, the file is deleted as well. If hard links to a file exist in multiple directories, each directory references the same inode number to access the file. No new inodes are created for hard links.

To create a hard link, use the following command:

```
ln <file name> hard link
```

Figure 15.20 displays the creation of a hard link.



The screenshot shows a terminal window titled "Caesor@localhost:~/Documents". The window contains the following text:

```
[Caesor@localhost Documents]$ ln doc1.txt doclinked
[Caesor@localhost Documents]$ ls
doc1.txt doclinked file2.txt file3.txt file4.txt file5.txt test1
[Caesor@localhost Documents]$
```

Figure 15.20: Creation of Hard Link

On the other hand, symbolic link is similar to a shortcut to a file on the file system. If the original file is deleted, the shortcut is no longer valid or becomes orphan. However, if the symbolic link is deleted, the original file is retained as is.

The syntax for the symbolic link is as follows:

```
ln -s <filename> symbolic link name
```

Figure 15.21 displays symbolic link.

```
[Caesor@localhost Documents]$ ln -s doc1.txt doc_short
[Caesor@localhost Documents]$ ls -l
total 36
-rw-rw-r--. 2 Caesor Caesor 8838 Apr 17 02:10 doc1.txt
-rw-rw-r--. 2 Caesor Caesor 8838 Apr 17 02:10 doclinked
lrwxrwxrwx. 1 Caesor Caesor     8 Apr 29 06:30 doc_short -> doc1.txt
-rw-rw-r--. 1 Caesor Caesor     0 Apr 16 09:35 file2.txt
-rw-rw-r--. 1 Caesor Caesor     0 Apr 16 09:35 file3.txt
-rw-rw-r--. 1 Caesor Caesor     0 Apr 16 09:35 file4.txt
-rw-rw-r--. 1 Caesor Caesor     0 Apr 16 09:35 file5.txt
-rw-rw-r--. 1 Caesor Caesor 8838 Apr 28 20:10 test1
[Caesor@localhost Documents]$
```

Figure 15.21: Symbolic Link

Transfer Files between Systems

A user can transfer files by using either the **ftp** command or the **scp** command. The **ftp** command sends the information to the destination server in an unencrypted form but the **scp** command, which is a replacement for **rcp** command, secures the information during the transfer.

The advantage of using the **scp** command is that it uses ssh protocol for securing the data transfers. It utilizes the security mechanism of the **ssh** protocol and authenticates users with the passwords.

The syntax for the **scp** command is as follows:

```
scp [-pqrvBC46] [-F ssh_config] [-S program] [-P port] [-c cipher] [-i identity_file] [-o ssh_option] [[user@] host1 : file1] [...] [[user@] host2 : file2]
```

The key parameters in this command are as follows:

```
scp <local-file> <username>@remote.domain.com:<remote-file>
```

The sample command is as follows:

```
scp data1.txt caesor@example.com:data1.txt
```

15.4.2 Removable Media

RHEL supports removable media. Some examples of removable media are CD, DVD, floppy and USB. A removable media must be mounted before it can be used. Mounting allows the removable media to be accessible on the RHEL operating system.

GNOME and KDE automatically mount the CD and DVDs. Figure 15.22 displays the mounted DVD.

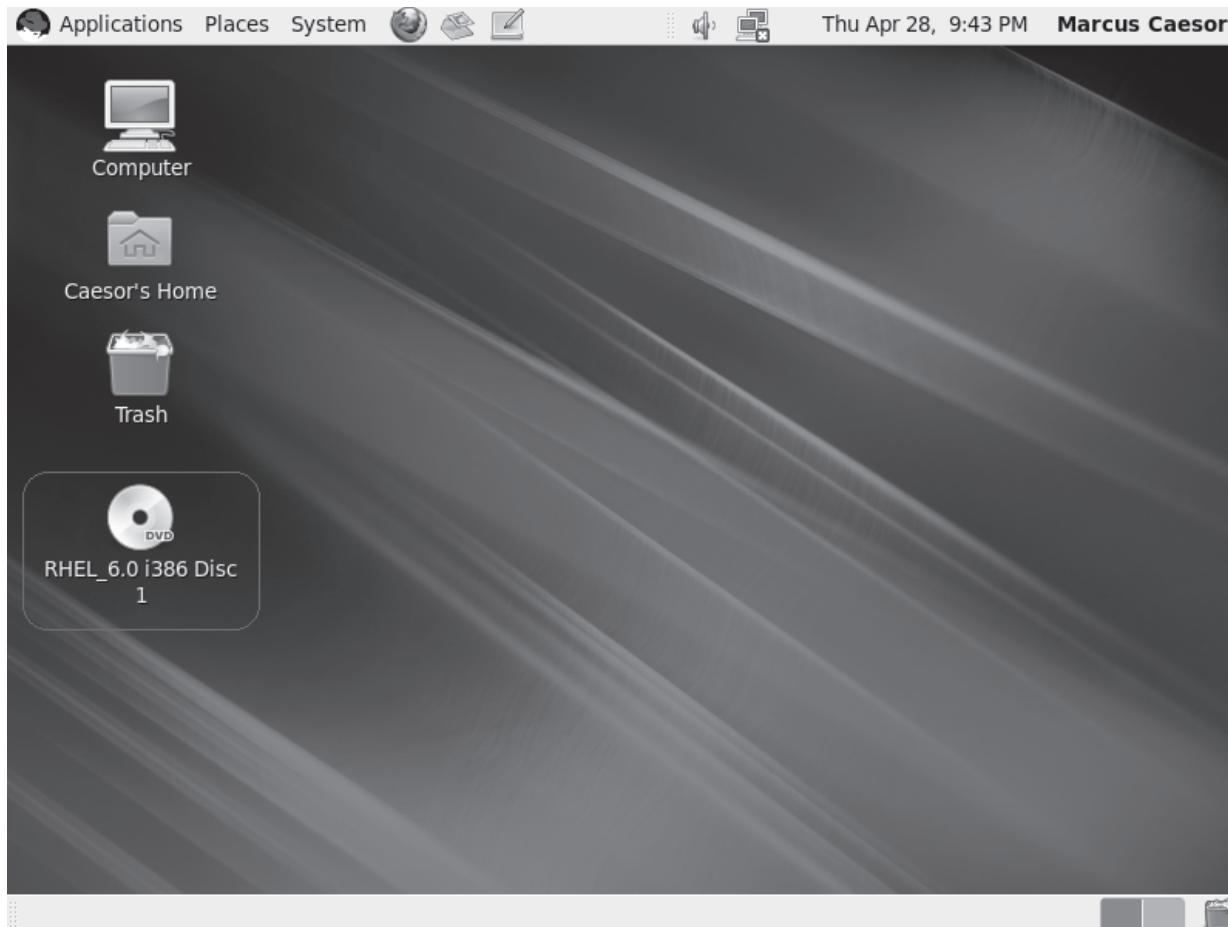


Figure 15.22: Mounted DVD

The mount points for the removable media are stored under the **/media** directory. In Figure 15.23, a DVD is mounted.

Figure 15.23 displays the mount point for the DVD when users browse the **/media** folder.

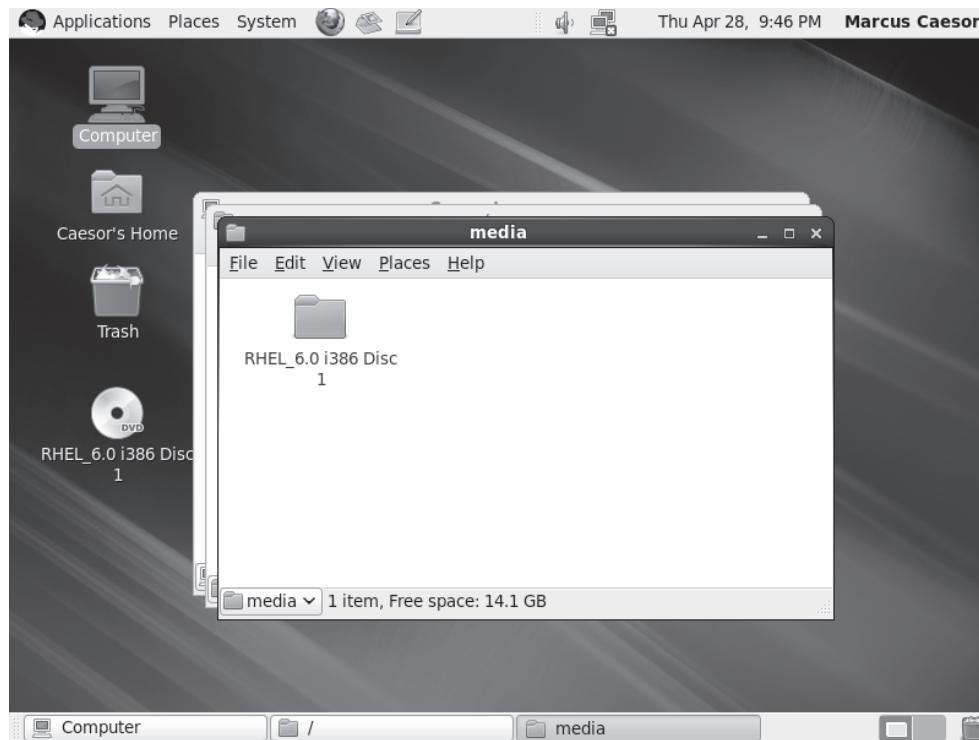


Figure 15.23: Mount Point

USB devices are detected as SCSI devices by the Linux kernel. The detected USB devices are typically labelled as sdaX or sdbX. The **/media** directory contains the USB device information, as in a DVD or CD.

USB devices are listed in the **/media** directory using the device ID. Floppies must be manually mounted or dismounted.

15.5 Security Fundamentals

RHEL, being a multiuser operating system, allow users to create and manage a user account for each user. To ensure security of the system, a system administrator assign policies for controlling the rights and permissions assigned to a user.

The various factors taken into account while creating these policies are as follows:

- The login ID and password that should be assigned to a user.
- The group to which a user belongs.
- The services to which a user can have an access.

These policies are used during the creation and administration of the user accounts. For example, a system administrator can assign policies for granting restricted permissions to RHEL users on the server to maintain the server security and integrity.

By default, the RHEL users have minimum permissions assigned to them. For the system level task, such as accessing log files or making changes to any configuration file, users are required to have root permissions.

15.5.1 Using Policies

Policies can also be used to restrict users from logging on to a specific system or disabling a user account after certain number of incorrect log in attempts.

The two types of policies considered while creating user accounts are as follows:

- User Account Policies
- Password Aging Policies

User Account Policies

Each user has a different account. The user account has unique properties or characteristics, such as the user ID, password and access permissions.

The factors that must be taken into account while creating a user account are as follows:

- User access to the system files and resources
- Periodic password changes required by the users for security reasons
- Logins would remain active
- CPU and memory limits should be allocated
- **Disk quota should be enabled**

All these factors will vary from system to system, depending on the importance of the data and resources attached to the system. For example, assume that all the employees in an organization are granted permission to take regular backups of their files to avoid accidental loss of data. However, the employees take backups of their personal data on the RHEL server, which consumes extensive disk space on the RHEL server. Hence, the policy should ensure that only a limited space is assigned to a user for taking the backups so that the disk space is utilized appropriately.

This is a situation when the user account is local to the system. Typically, the domain level accounts and the policies are applied to a group of users, which are normally in an organizational unit.

It is recommended that an organization should devise a strict policy that gives only the required permissions to users. A lenient policy that grants maximum permissions to the users, can lead to security issues, such as accessing the confidential data.

The following factors are considered when assigning login IDs and passwords to users:

- The login IDs must be unique. The login ID can be decided based on the size, structure or nature of the organization.
- The passwords should be at least eight characters in length. Short passwords are weak passwords and can be easily guessed. For example, P00r and ph2004 are weak passwords.
- The passwords should not be easy to decipher. Passwords should be based on an expanded character set, which includes mixed case, alphanumeric characters. For example, P@ss5w0rd.

Password Aging Policies

The password aging policy sets the password age for a user account. In Linux, the password does not expire by default. Therefore, using the password aging policy, the user must set the duration for which a password remains valid. At the end of that duration, the user will be prompted by the operating system to enter a new password. This new password remains valid only till the expiry date of the password specified for the user account.

A password set for a longer duration provides less security as the password can get leaked to other users. Therefore, it is advisable to change the password after short durations. For example, the passwords should be changed within 15 days. To configure the password settings, the **chage** command can be used.

The syntax for the **chage** command is as follows:

```
chage [-m mindays] [-M maxdays] [-d lastday] [-I inactive] [-E expiredate] [-W warndays] user
```

Table 15.6 lists the options available with the chage command.

Option	Function
-m <mindays>	Specifies the minimum number of days within which a user has to change the password. If the value of the –m parameter is set to zero, it indicates that the password would never expire and the user can change the password at any time.
-M <maxdays>	Specifies the maximum validity period (in days) of the password.
-d <lastday>	Specifies the number of days, when the user last changed the password.
-I <inactive>	Specifies the number of days an account is inactive. An account is inactive after a password has expired and before the account is locked.
-E <expiredate>	Sets the date in the YYYY-MM-DD format. After the specified date the account would no longer be accessible by the user.
-W <warndays>	Sets the number of days when a warning would be displayed to intimate the user that the password change is required.

Table 15.6: Options of **chage** Command

For example, the user has to set the age of a user account's password to seven days. The following command can be used for this purpose:

```
# chage -m 7 susan
```

In the preceding command, the user has set the minimum number of days within which the password has to be changed for the user name, Susan, to 7. Only the root user can change the password settings of other users by using the **chage** command. However, if a user has to determine when the user's password is going to expire, the user can use the **chage** command with the following syntax:

```
chage -l <user-name>
```

The following command displays all the values set for xyz's password:

```
# chage -l xyz
```

The output of the preceding command displays the password details as shown in Figure 15.24.

```
[xyz@Marcus ~]$ chage -l xyz
Last password change : May 27, 2011
Password expires : never
Password inactive : never
Account expires : never
Minimum number of days between password change : 0
Maximum number of days between password change : 99999
Number of days of warning before password expires : 7
[xyz@Marcus ~]$
```

Figure 15.24: Password Details

The preceding command displays the values for each parameter, such as minimum and maximum days set for the user Susan. Here, the warning parameter specifies that the user would receive password expiration warning messages five days before the password expiry date. The inactive parameter mentions the number of days since the password was inactive.

15.5.2 Authenticating Users

When a user logs on to the Linux operating system, the operating system authenticates the ID and password entered by the user. This authentication can take place on the local system or on the user database stored on the remote server, which is typically a directory server holding all user accounts of an organization.

Local authentication is implemented using the shadow password and Message Digest5 (MD5) password schemes. Network authentication is implemented using the Network Information System (NIS), Lightweight Directory Access Protocol (LDAP) or SMB authentication schemes. These authentication

schemes can be selected during the installation of the operating system or after the installation with the help of the `authconfig` command.

→ Using Local Authentication

Linux provides password options, such as shadow password and MD5 password, to implement local authentication. In Linux, when a new user account is created, an entry for the user is reflected in the user database file `/etc/passwd`. This file contains fields, including a password field for each user. The `/etc/passwd` is a readable file that can be easily changed or can be used to trap the password for any user.

Linux provides a shadow password scheme to encrypt the passwords. If the shadow password scheme is enabled, the passwords that are encrypted are stored in the `/etc/shadow` file. Only the root user can read these passwords. The `/etc/shadow` file stores one record for each user account created on the Linux system.

The owner of the `/etc/shadow` file is a root user and the root user can change the file permissions, it is recommended not to do so because this can lead to a possible compromise of the whole system.

The format of the `/etc/shadow` file is as follows:

`username:passwd:last:may:must:warn:expire:disable:reserved`

Table 15.7 lists the fields of the `/etc/shadow` file.

Field	Description
username	Describes the user login name
password	Describes the password in the encrypted form
last	Describes the number of days, the current date and the date when the password was last changed
may	Describes the number of days before which the password can or cannot be changed
must	Describes the number of days post which the password must be changed
warn	Describes the number of days left for the password to expire, so that the user is warned
expire	Describes the number of days after which the password expires and the account is disabled
disable	Describes the number of days between the current date and the date on which the account was disabled
reserved	Is a reserved field that can be used in future

Table 15.7: Fields of the `/etc/shadow` File

The shadow password scheme secures the system from intrusions and nullifies the chances of unauthorized users using the system.

15.6 Check Your Progress

1. Which of the following **su** commands is used to switch to the home directory?

(A)	su root	(C)	su root<password>
(B)	su - root	(D)	su - root<password>

2. In which of the following directories is the log for the **su** command maintained?

(A)	/var/adm/syslog	(C)	/var/syslog
(B)	/root/log	(D)	/var/system/log

3. Which of the following file is used for maintaining the UIDs of users?

(A)	/etc/Userpwd	(C)	/var/passwd
(B)	/etc/passwd	(D)	/var/UIDs

4. Which are the default members of the root group?

(A)	root, bin, daemon	(C)	root, bin
(B)	root	(D)	root, daemon, system

5. What is the range in which system accounts exist?

(A)	1-99	(C)	1-400
(B)	1-100	(D)	1-499

6. Which parameter is used with **useradd** command to create the user home directory?

(A)	-m	(C)	-n
(B)	-M	(D)	-d

7. Which tab in the User Manager is used to set the password expiration?

(A)	User Data	(C)	Password Info
(B)	Account Info	(D)	User Expiration

8. Which tool is used to create users in the graphical environment?

(A)	Users and Groups	(C)	User Manager
(B)	Administration	(D)	Account Manager

9. Which of the following is the default directory in which user's home directory is created?

(A)	/home	(C)	/var
(B)	/etc	(D)	/root

10. A user account has to be created without the home directory. Which parameter must be used with the **useradd** command?

(A)	-m	(C)	-n
(B)	-M	(D)	-d

11. Which parameters are used with the **rm** command to remove a directory with files?

(A)	-m	(C)	-r
(B)	-M	(D)	-d

12. Which commands securely transfer files between two computers?

(A)	fcp	(C)	mv
(B)	cp	(D)	scp

13. What would be the outcome of the `cp` command?

(A)	A new directory with the same name would be created and the file would be copied to that directory.	(C)	The copy operation would fail.
(B)	A file with the directory name would be created.	(D)	The file would be copied to the source directory itself as a duplicate file.

14. Which of the following is a shortcut to a file in another directory?

(A)	hard link	(C)	inode
(B)	symbolic link	(D)	dentry

15. Which command is used to create a symbolic link?

(A)	<code>ln -s</code>	(C)	<code>ln -d</code>
(B)	<code>ln -l</code>	(D)	<code>ln -m</code>

16. Which of the following commands are used to configure the password settings?

(A)	<code>chage</code>	(C)	<code>mkdir</code>
(B)	<code>chmod</code>	(D)	<code>authconfig</code>

17. Which of the following options of the `chage` command specifies minimum number of days within which a user has to change the password?

(A)	<code>-M</code>	(C)	<code>-d</code>
(B)	<code>-m</code>	(D)	<code>-w</code>

18. Which of the following options of the chage command sets the number of days to display a warning before a password change is required?

(A)	<code>-m</code>	(C)	<code>-d</code>
(B)	<code>-M</code>	(D)	<code>-w</code>

19. Which of the following commands starts a child shell?

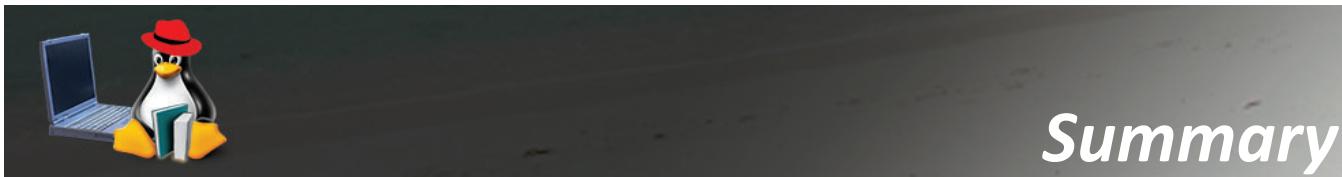
(A) chage	(C) su
(B) mkdir	(D) sudo

20. Which of the following commands enables a non-root user to use root privilege?

(A) chage	(C) su
(B) mkdir	(D) sudo

15.6.1 Answers

1.	B
2.	A
3.	B
4.	B
5.	D
6.	A
7.	B
8.	C
9.	A
10.	B
11.	C
12.	D
13.	B
14.	B
15.	A
16.	A
17.	B
18.	D
19.	C
20.	D



- To access the RHEL environment a user account is required. Multiple user accounts can be created on a single RHEL system. The user accounts provide an identity for the user. These identities can be changed in different ways such as changing from a user account to root user account or changing from root user account to another user account. The root user has complete control over the RHEL environment. The differences between a root and a normal user account depend on the UIDs of the root and normal user accounts. There is a process to create user accounts and check its properties along with the requirement to create a group by combining multiple users.
- User accounts are created using the User Manager. The user account properties of the user accounts can also be edited in the User Manager. A user account can also be created from the command-line interface. To create user account from the command-line interface, the useradd command with various parameters can be executed.
- Users are often required to copy or move files from one directory to another directory. In copy operation, the source file is retained as is and another copy of the source file is created at the defined destination. In a move operation, the source file is deleted from the source location and moved to the destination location. Advanced operations include dentries and inodes, and symbolic and hard links.
- The scp command can be used to securely transfer files from one system to another system. Users can transfer files in secure manner using the scp command.
- Users can use removable media on RHEL. Some examples of removable media are CD, DVD, floppy, and USB. A user must mount the removable media before it can be used.
- To ensure security of the system, a system administrator prepares policies for controlling the rights and the permissions assigned to the user. These policies are used when creating and administering the user accounts. The two types of policies considered while creating user accounts are User account policies and Password aging policies.
- Local authentication is implemented using the Shadow Password and Message Digest 5 (MD5) Password schemes. Network authentication is implemented using the Network Information System (NIS), Lightweight Directory Access Protocol (LDAP) or Session Message Block (SMB) authentication schemes. The account of a new user is locked by default and RHEL enables users to assign one user's privileges to another user.



GLOSSARY

look-up guide

Terms Acronyms Programming

Animation Aviation Graphics

Exercise - 1**Changing identities**

A user can change identity while working and act like another user. This is useful because a user does not have to log off from the current user account to perform an action that only another user can perform. For example, a user has to open the directory where log files are stored. However, when the user attempts to open the directory, the user is prompted with permission denied error. In this case, the user can use the **su** command to become a root user and then access the directory with the log files.

Note - In this lab, there must be another user, Marcus.

Step 1 - Log on to the RHEL system with the login credentials.

Step 2 - Click **Applications** → **System Tools** → **Terminal** to open terminal.

Step 3 - Execute the **cd..** command.

The command directs to the home folder where the home directories for all users are located, as shown in Figure 16.1.

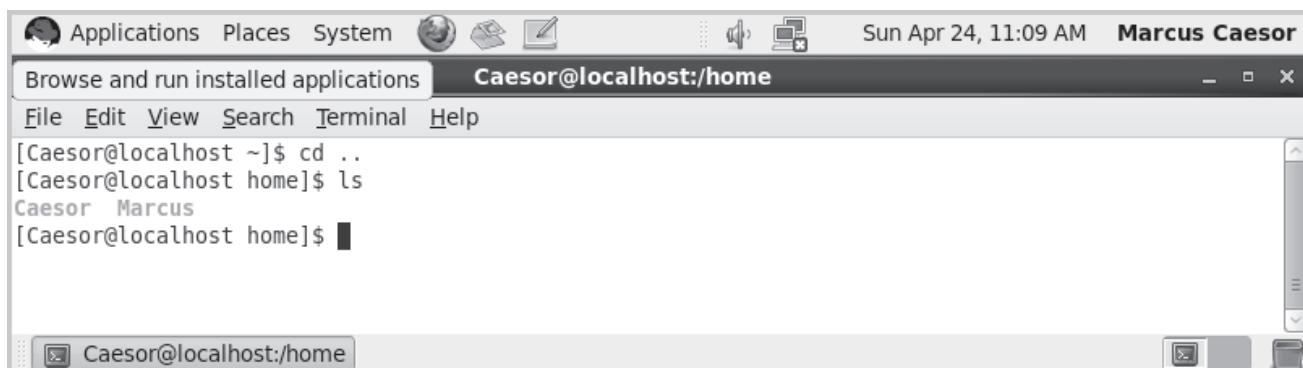


Figure 16.1: Home Directory

Step 4 - Run the following command:

```
# cd Marcus
```

The user gets permission denied error while attempting to do this..

Step 5 - The current user has to change identities similar to a root user identity to access this directory.

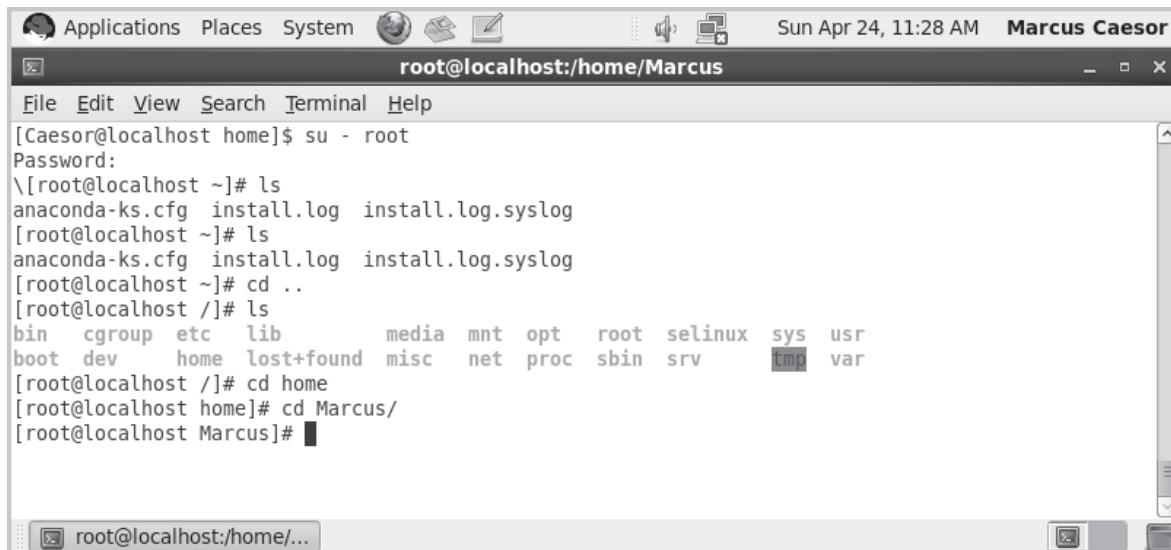
Run the **su - root** command.

Step 6 - At the command prompt for password for the root user, enter the password.

Step 7 - Type **cd..** to go to the previous directory and then, navigate to the user's home directory.

Step 8 - Run the **cd Marcus** command.

As a root user identity, the user is able to navigate to other user's home directory as shown in Figure 16.2.



The screenshot shows a terminal window titled "root@localhost:/home/Marcus". The window is part of a desktop environment with a menu bar at the top. The terminal content shows the user switching to root, navigating through directories, and listing files. The command history includes:

```
[Caesor@localhost ~]$ su - root  
Password:  
\[root@localhost ~]\# ls  
anaconda-ks.cfg install.log install.log.syslog  
\[root@localhost ~]\# ls  
anaconda-ks.cfg install.log install.log.syslog  
\[root@localhost ~]\# cd ..  
\[root@localhost /]\# ls  
bin cgroup etc lib media mnt opt root selinux sys usr  
boot dev home lost+found misc net proc sbin srv tmp var  
\[root@localhost /]\# cd home  
\[root@localhost home]\# cd Marcus/  
\[root@localhost Marcus]\#
```

Figure 16.2: Root User Identity

Step 9 - Type **exit** at the command prompt and press **ENTER** to exit from the **su** command.

Step 10 - Type **exit** and press **ENTER** to exit from the command prompt.

The shell is closed and desktop is displayed.

Exercise - 2

Creating new users

Step 1 - Click **System → Administration → Users and Groups** to open User Manager.

Step 2 - Click Add User. Figure 16.3 shows adding a new user.

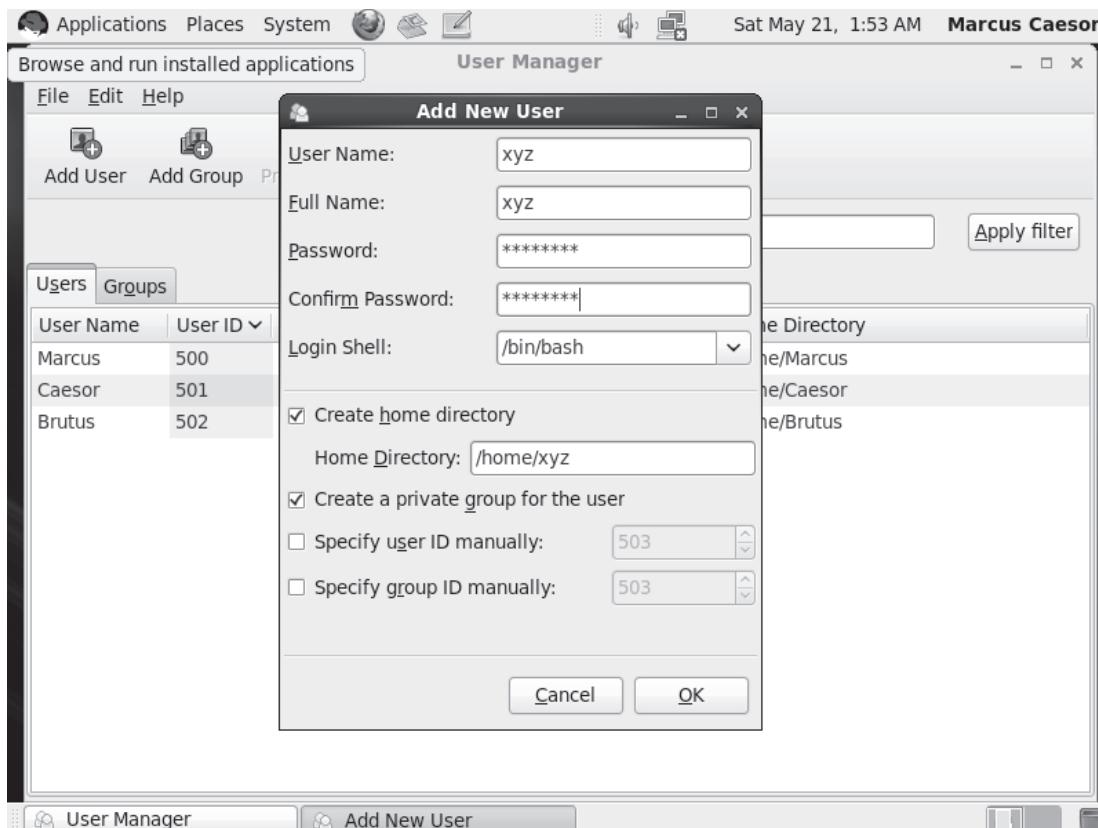


Figure 16.3: Adding a New User

Step 3 - In the User Name box, type the user name.

Step 4 - In the Full Name box, type the full name of the user.

Step 5 - In the Password box, type the user password.

Step 6 - In the Confirm Password box, retype the user password.

Note - RHEL does not accept the dictionary-based passwords. It is necessary to define complex passwords that consist of alphanumeric and special characters and are longer in length. An example is P@\$\$w0rd!123.

Step 7 - In the Login Shell list, select the default login shell, /bin/bash.

Step 8 - The default path for the home directory is /home/username. Clear the Create home directory check box if the home directory is not required.

Step 9 - The Create a private group for the user check box is selected by default. Keep the selection.

Step 10 - It is also possible to manually define the user ID and group ID. Skip them and click OK (Refer to Figure 16.4).

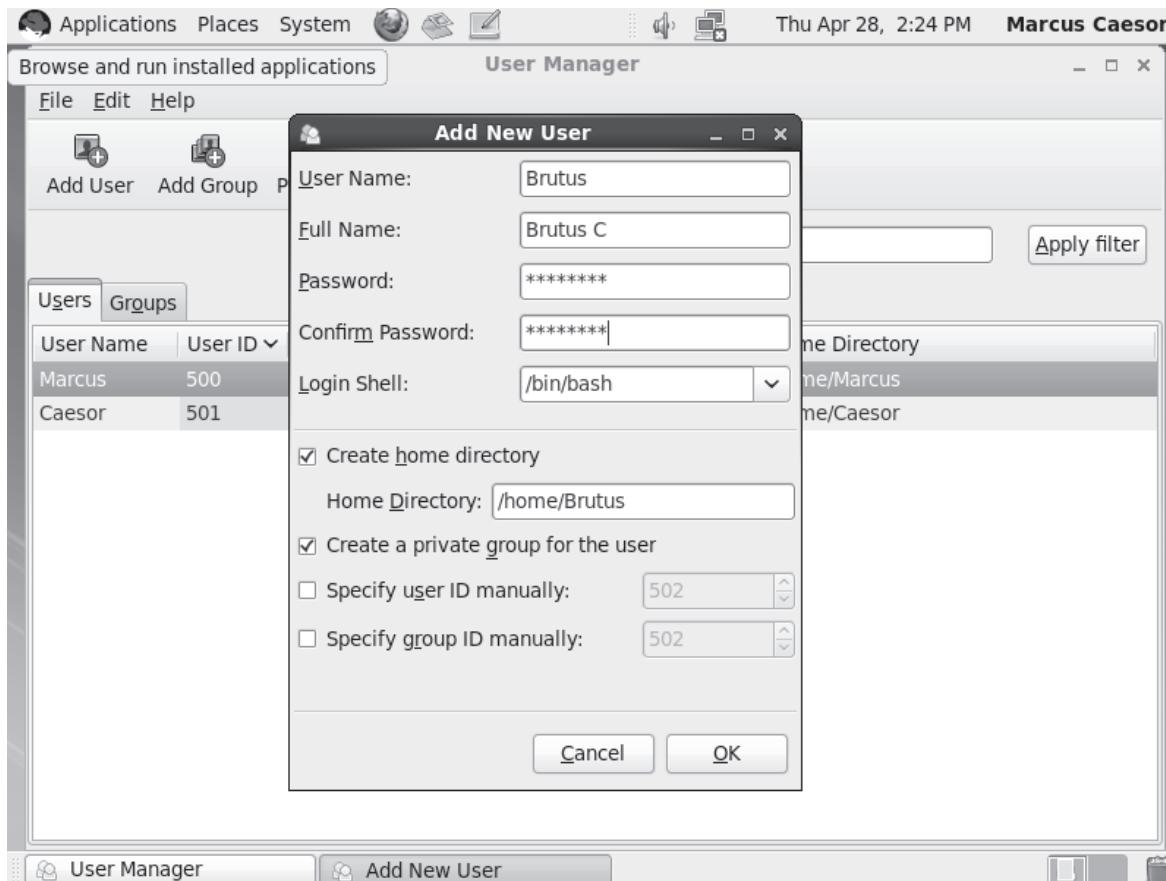


Figure 16.4: Creating a User

Note - After creating the user, the superuser can modify to configure the advanced properties, such as password expiration on the Account Info tab of the user properties.

To modify the user settings, select the user and click Properties.

Figure 16.5 displays the Properties for a user.

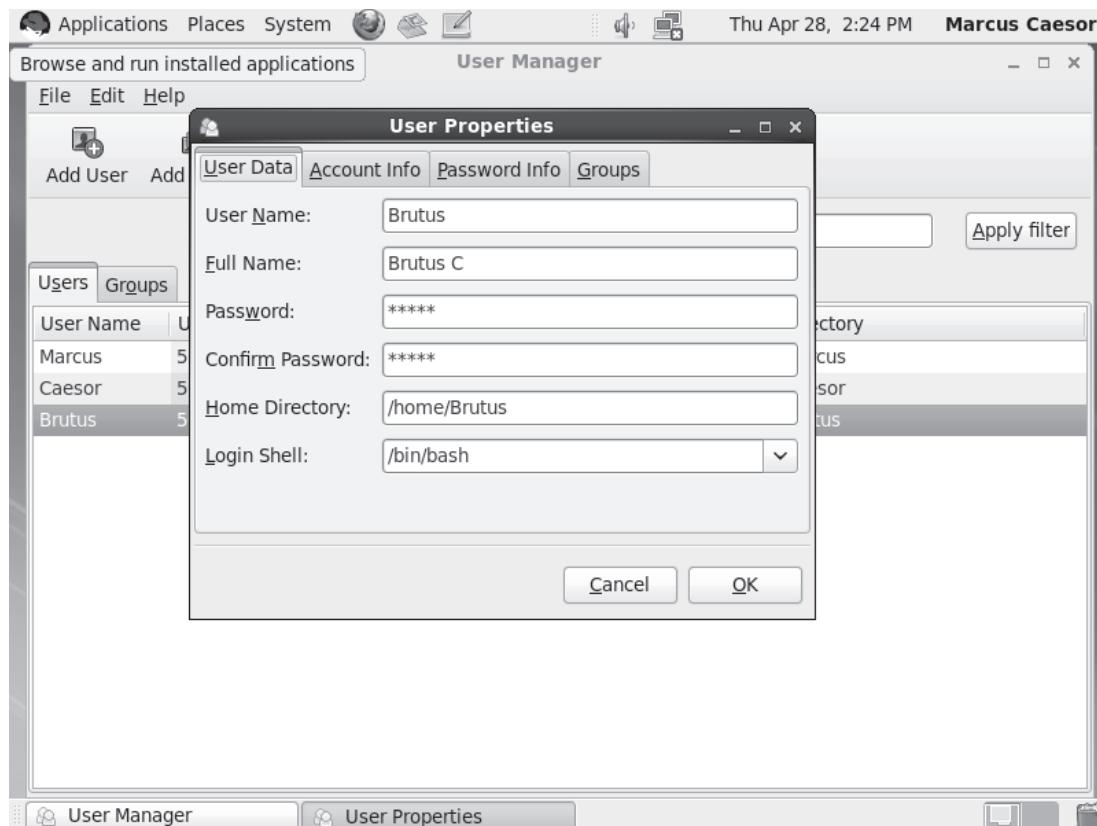


Figure 16.5: User Properties Dialog Box

Exercise - 3

Creating a group using the graphical environment

Step 1 - Log on to the RHEL system with the login credentials.

Step 2 - To open User Manager, Click **System → Administration → Users and Groups**.

Step 3 - Type the root user password for authentication purpose as the user account is not authorized for modifying any user account (Refer to Figure 16.6).

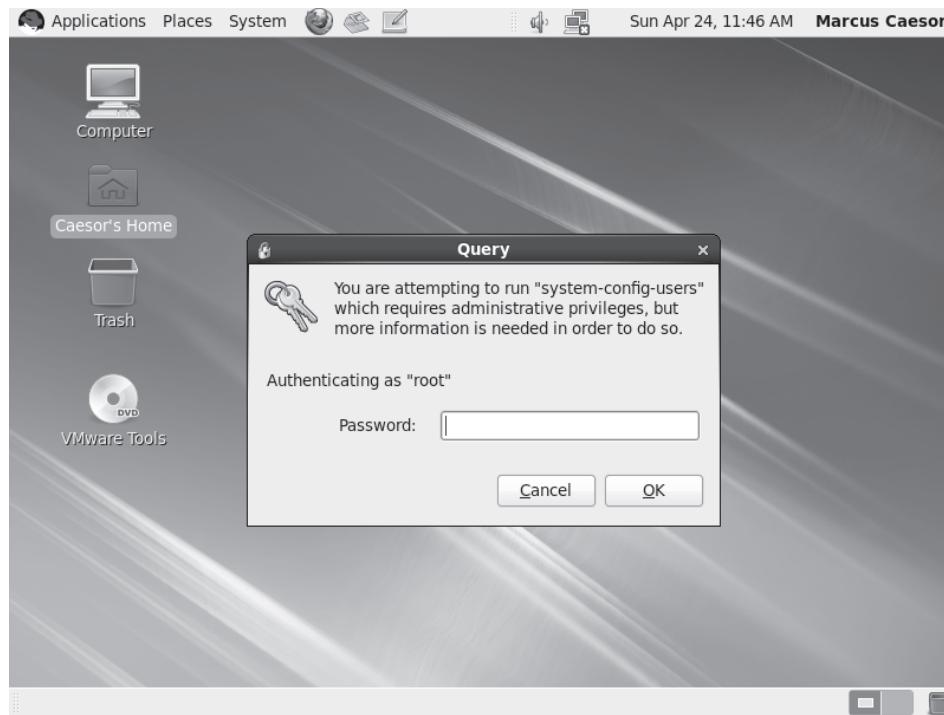


Figure 16.6: Root User Password Prompt

Step 4 - Click OK. The system displays the User Manager window as shown in Figure 16.7.

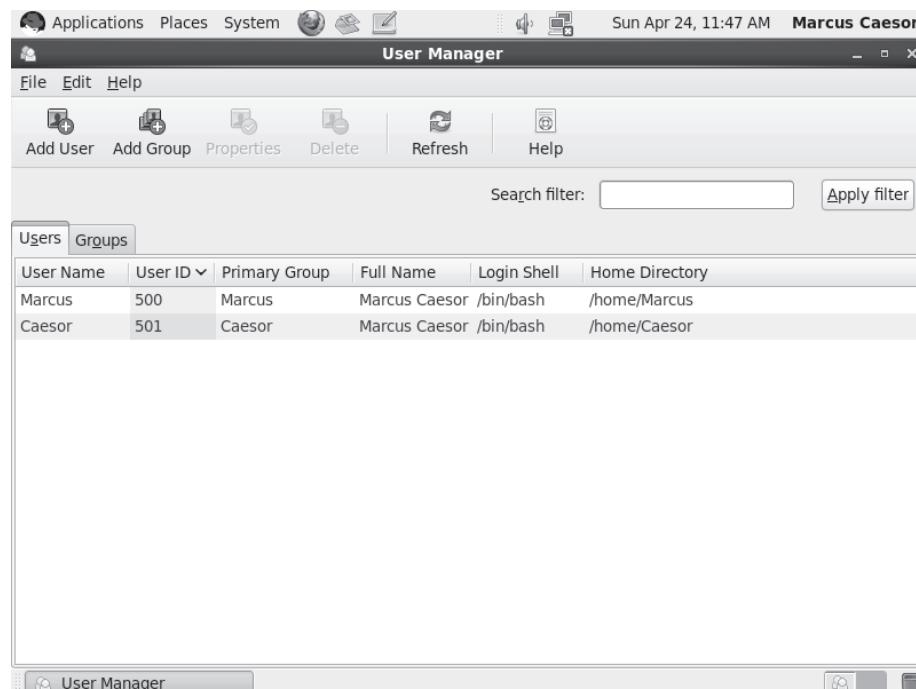


Figure 16.7: User Manager Window

Step 5 - Click Add User. The system displays the Add New User dialog box as shown in Figure 16.8.

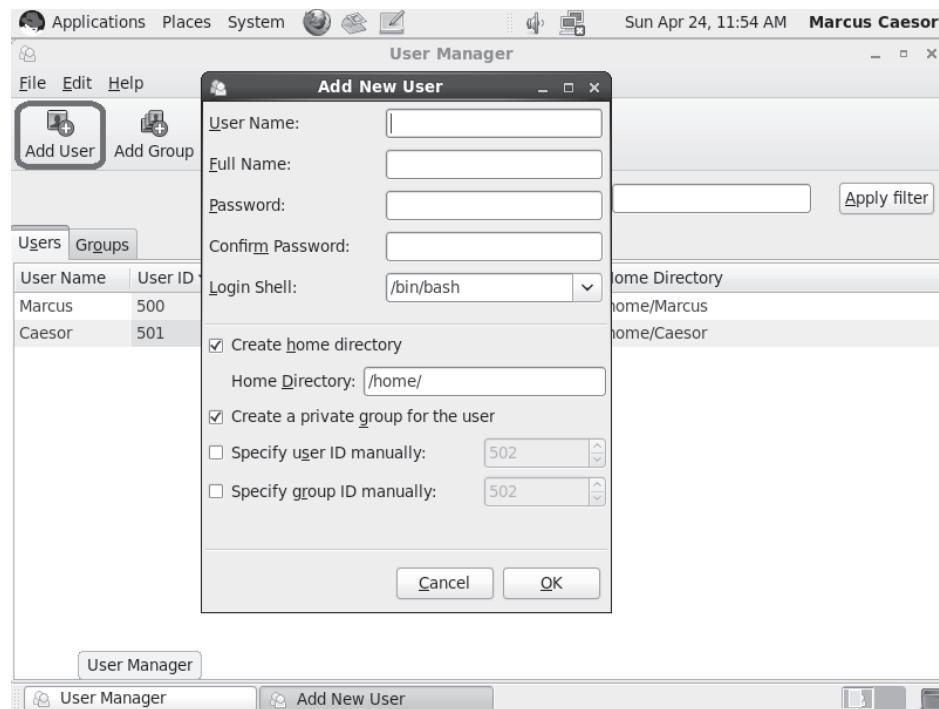


Figure 16.8: Add New User Dialog Box

Step 6 - Fill in the required fields and click OK (Refer to Figure 16.9).

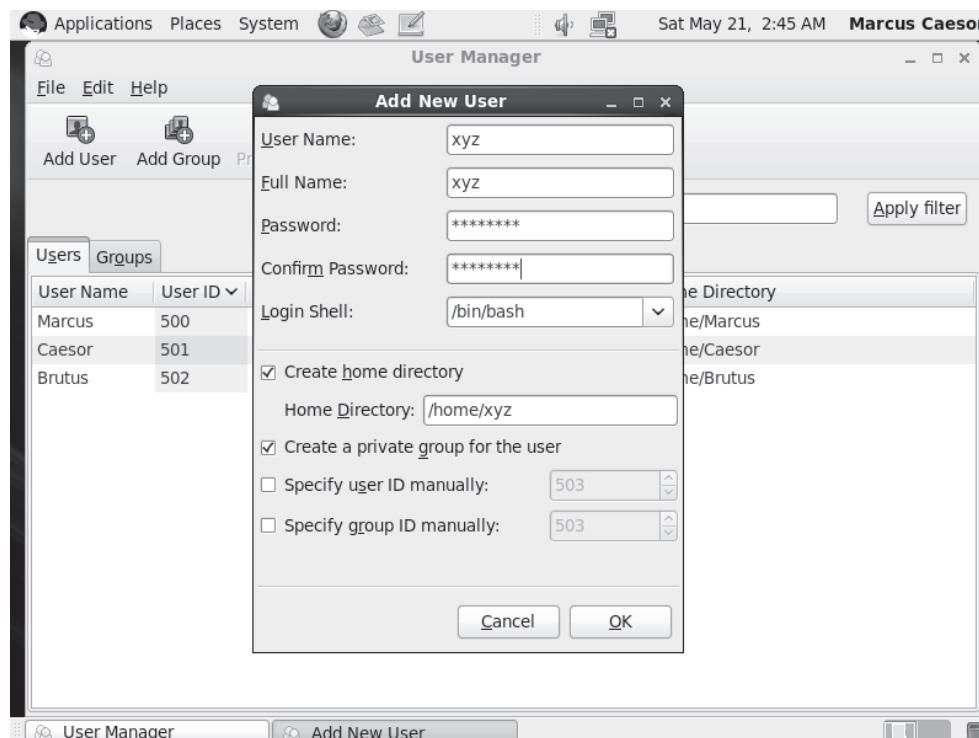


Figure 16.9: Adding User Details

Note - If a weak password is defined, the system displays a message. Click Yes to continue with the weak password.

Step 7 - A new user is defined in the system with the name that was defined in the **Add New User** dialog box. To define more details, in the **User Manager** windows, in the Users tab, select the user and click Properties. The User Properties is displayed as shown in Figure 16.10.

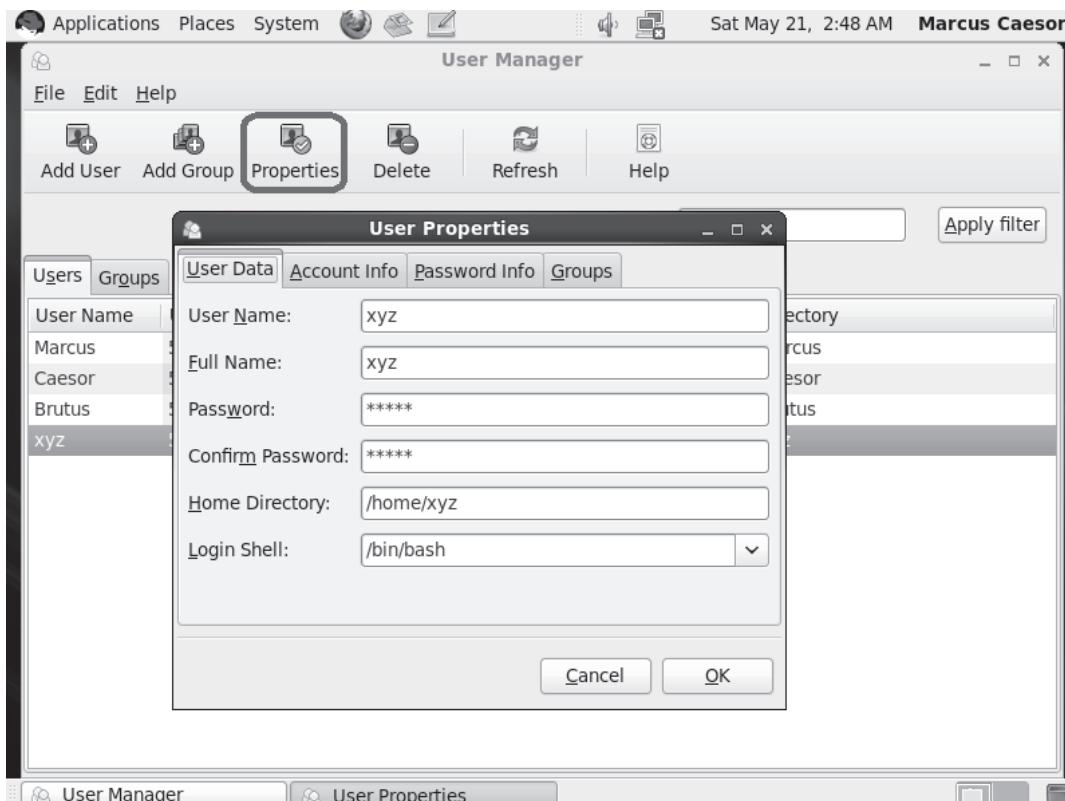


Figure 16.10: Entering User Properties

Note - In the User Properties dialog box, configure settings such as set the user group, set the password expiration policies and enable password expiration.

Step 8 - Modify the details as required and click **OK** to close the User Properties dialog box.

Step 9 - Exit User Manager.

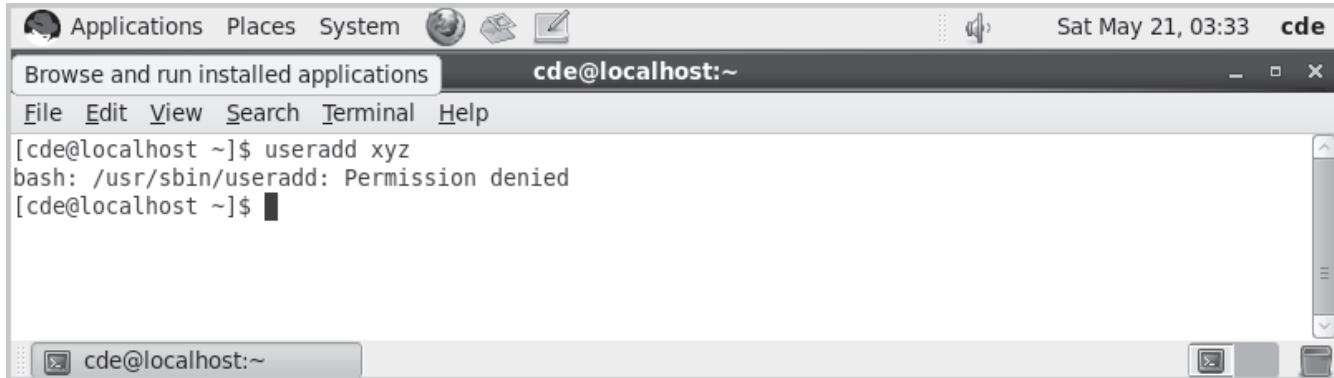
Exercise - 4

Creating a group using the command-line interface

Step 1 - Open the command prompt from **Applications → System Tools → Terminal**.

Step 2 - Execute the **useradd xyz** command.

Figure 16.11 shows the permission denied error.



The screenshot shows a terminal window titled 'cde@localhost:~'. The window contains the following text:

```
[cde@localhost ~]$ useradd xyz
bash: /usr/sbin/useradd: Permission denied
[cde@localhost ~]$
```

Figure 16.11: Permission Denied Error

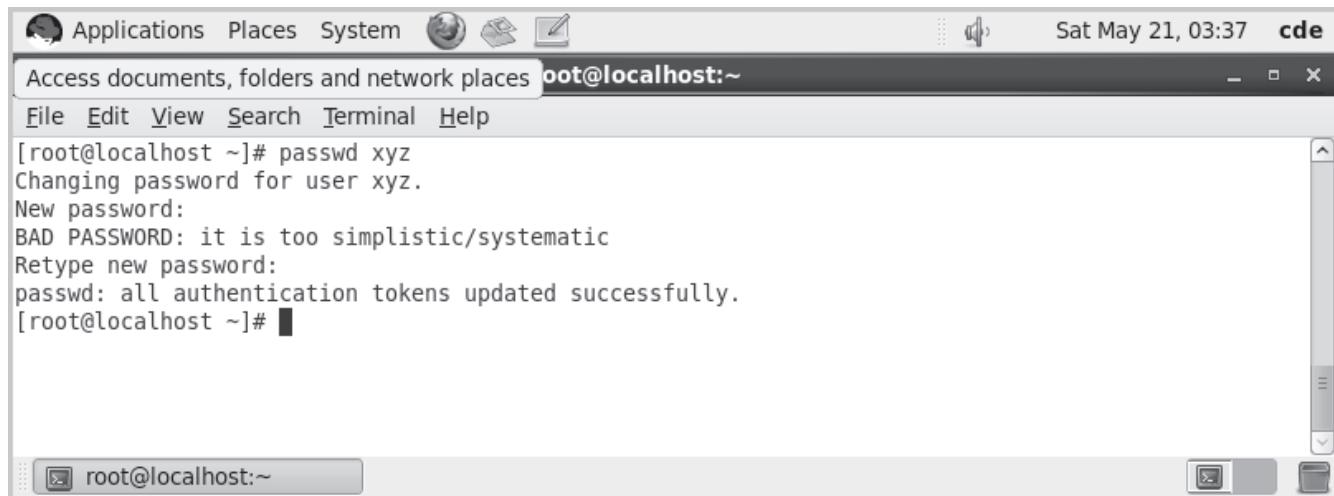
Note - To add users, a user must log in as a root user. The **su** command can be used to log in as a root user.

Step 3 - Log in as a root user using the **su** command and provide the password for the root user.

Step 4 - Execute the **useradd xyz** command to create a user.

A new user is created. Next, define the password for the user.

Step 5 - Enter a password, and then confirm the password for the user xyz as shown in Figure 16.12.



The screenshot shows a terminal window titled 'root@localhost:~'. The window contains the following text:

```
[root@localhost ~]# passwd xyz
Changing password for user xyz.
New password:
BAD PASSWORD: it is too simplistic/systematic
Retype new password:
passwd: all authentication tokens updated successfully.
[root@localhost ~]#
```

Figure 16.12: Creating Password for a New User

Step 6 - To verify whether the new user is successfully created, the users that exist in the system must be listed. To generate the list of existing users, execute the following command:

```
cat /etc/passwd | grep 500*
```

Figure 16.13 displays the users that have UID more than 500.

```
[root@localhost ~]# cat /etc/passwd | grep 500*
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
abc:x:501:501:abc:/home/abc:/bin/bash
cde:x:502:502:cde:/home/cde:/bin/bash
xyz:x:503:503::/home/xyz:/bin/bash
[root@localhost ~]#
```

Figure 16.13: Users Listed in passwd File

Exercise - 5

Creating a Group

Step 1 - At the command prompt, execute **groupadd aaa** command.

Step 2 - To add an existing user to the aaa group, execute the **usermod -a -G aaa xyz** command.

Step 3 - To verify that the user marcus is added to the marketing group, execute the **id xyz** command.

The result of the **id xyz** command is shown in Figure 16.14.

```
[root@localhost ~]# usermod -a -G marketing marcus
[root@localhost ~]# id marcus
uid=504(marcus) gid=504(marcus) groups=504(marcus),505(marketing)
[root@localhost ~]#
```

Figure 16.14: Adding Existing User to a Group

Step 4 - Type **exit** to log out of the **su** command.

Exercise - 6

Allowing a user to change password

Step 1 - At the command prompt, type **passwd**.

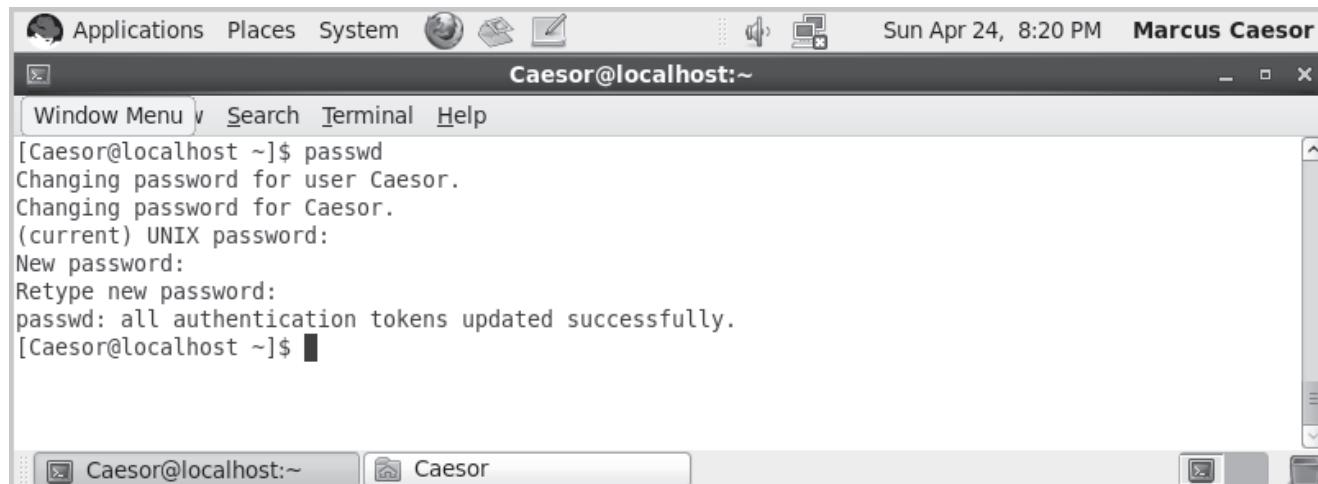
The current user password is prompted.

Step 2 - Type the current password.

Step 3 - Type a new password.

Step 4 - Confirm the new password by retyping it.

If both the passwords match, the system displays a message that authentication tokens are successfully updated (Refer to Figure 16.15).



The screenshot shows a terminal window titled "Caesor@localhost:~". The window contains the following text:

```
[Caesor@localhost ~]$ passwd
Changing password for user Caesor.
Changing password for Caesor.
(current) UNIX password:
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[Caesor@localhost ~]$
```

Figure 16.15: Password Changed Successfully

Note - Users can change the password for themselves but not for other users. Users can log on as root user using the **su** command, to change the passwords for other users. The command to change the password for other users, when logged in as root user, is **passwd <username>**.



To add on to your knowledge of the subject,
visit the **REFERENCES** page

