Red Hat Enterprise Linux 8.0 RH124 - Red Hat System Administration I

Chapter 1: Getting Started with Red Hat Enterprise Linux

Key Concepts:

- **Open Source**: Software where the source code is available and typically licensed to allow users to study, change, and distribute it. Benefits include the code surviving the loss of the original developer and the ability for users to learn from and develop upon the code.
- **Linux**: An operating system kernel, defined and explained in purpose. Includes a powerful and scriptable command-line interface useful for automation and provisioning.
- **Linux Distribution**: A complete operating system built around the Linux kernel, bundling it with other software like system utilities, libraries, and applications. Purpose is defined and explained.
- **Red Hat Enterprise Linux (RHEL)**: Red Hat's specific Linux distribution that is open source, designed for enterprise use, and includes commercial support. Its purpose is defined and explained.

Important Commands:

- No specific commands were highlighted as central to the introductory concepts in the provided sources for this chapter.

Applications:

- Understanding these foundational concepts is the starting point for system administration on RHEL.
- Recognizing the benefits of open source software can influence software choices and development practices.
- Appreciating the power of the Linux command line sets the stage for subsequent chapters focusing on command-line administration.

Chapter 2: Accessing the Command Line

Key Concepts:

- **Command Line**: A text-based interface used to interact with a computer system by typing instructions.
- **Shell**: A program that provides the command-line interface in Linux and interprets user commands. Different shell programs exist, but Bash is the current default for most users.
- **Bash Shell (GNU Bourne-Again Shell)**: A specific and commonly used shell program that interprets commands.
- **Command Syntax**: Commands generally consist of three parts: the **command** name, followed by **options** (usually starting with `-` or `--`), and **arguments** (often indicating targets). Parts are separated by spaces.

- **Accessing the Command Line**: Can be done via a local text console or from a terminal program within the GNOME 3 desktop environment.
- **Bash Shortcuts**: Features like tab completion, command history, and command editing can save time when working in the Bash shell.
    - **Tab completion**: Helps complete file names, commands, and options when typing.
    - **Command history**: Stores previously executed commands, allowing recall and re-execution.

Important Commands:

- **`history`**: Displays a numbered list of commands previously entered in the current session.
- **`!number`**: Executes the command from history corresponding to the specified number.
- **`!string`**: Executes the most recent command from history that starts with the specified string.
- **`Esc+.`** **(Escape key followed by period)**: Copies the last argument of the previous command to the current command line.
- **`date`**: Displays the current date and time. Can be used with format strings like `+%r` for 12-hour time.
- **`file`**: Determines the type of a file (e.g., text, executable, script).
- **`wc`**: Counts lines, words, and bytes in a file.
- **`head`**: Displays the first few lines of a file.
- **`tail`**: Displays the last few lines of a file. The `-n` option specifies the number of lines.

Applications:

- Logging into and interacting with a Linux system.
- Executing basic system commands for tasks like checking the time (`date`), inspecting file types (`file`), or viewing file content (`head`, `tail`).
- Increasing efficiency and reducing typing errors when entering commands using tab completion, history, and editing shortcuts.
- Performing simple administrative tasks and program execution from the command line.

Chapter 3: Managing Files From the Command Line

Key Concepts:

- **File-system Hierarchy**: The organization of files on a Linux system into a single inverted tree structure, with the root directory (`/`) at the top.
- **Paths**: Specify the location of a file or directory.
    - **Absolute paths**: Start with `/` and specify the location from the root of the file system.

- **Relative paths**: Do not start with `/` and specify the location relative to the current working directory.
- **Current Working Directory**: The directory the user is currently operating within.
- **Directory Navigation**: Moving between directories in the file system hierarchy.
- **File and Directory Management**: Creating, copying, moving, and removing files and directories are fundamental tasks.
- **Links**: Ways to create multiple directory entries (names) that point to the same file data.
    - **Hard Link**: A directory entry that refers directly to the file's data (inode). If the original name is deleted, the data remains accessible via the hard link until the last link is removed. Changes made via one link are seen via all.
    - **Soft Link (Symbolic Link)**: A directory entry that is a pointer to another file or directory by name. If the original file is deleted, the soft link becomes broken.
- **Bash Shell Expansions**: Features that allow the shell to automatically generate lists of file names or text strings, making commands more efficient, especially when dealing with multiple files.
    - **Pattern Matching (Globbing/Wildcards)**: Using special characters (metacharacters) to match file names.
        - `*`: Matches any sequence of characters (including none).
        - `?`: Matches any single character.
        - `[]`: Matches any single character within the brackets. Ranges can be specified (e.g., ``).
        - `{}`: Brace expansion generates lists of strings or sequences (e.g., `{1..6}` generates `1 2 3 4 5 6`; `{cat,dog}` generates `cat dog`).
    - **Home Directory Expansion**: `~` expands to the current user's home directory.
    - **Command Substitution**: Using the output of one command as arguments for another command.

Important Commands:

- **`pwd`**: Prints the absolute path of the current working directory.
- **`cd`**: Changes the current directory. `cd` or `cd ~` goes to the home directory. `cd ..` goes up one directory. `cd -` goes to the previous working directory.
- **`ls`**: Lists directory contents. Common options include `-l` (long listing), `-a` (all files, including hidden), `-R` (recursive).
- **`mkdir`**: Creates new directories.
- **`rmdir`**: Removes empty directories.
- **`touch`**: Creates new empty files or updates the timestamp of existing files.
- **`cp`**: Copies files or directories.
- **`mv`**: Moves or renames files or directories.
- **`rm`**: Removes files or directories.
- **`ln`**: Creates links. `ln target link_name` creates a hard link. `ln -s target link_name` creates a soft link.

Applications:

- Organizing files and directories into a logical structure.
- Navigating efficiently through the file system using absolute and relative paths.
- Creating, copying, moving, and deleting files and directories as part of system administration and user tasks.
- Creating backups or alternative access points to files using hard and soft links.
- Performing operations on multiple files efficiently using wildcards and brace expansion.
- Automating file naming or processing using command substitution.
- Saving command output to files for later use.

Chapter 4: Getting Help in Red Hat Enterprise Linux

Key Concepts:

- **Local Help Systems**: Documentation available directly on the RHEL system.
- **Manual Pages (Man Pages)**: Documentation for commands, files, system calls, etc., accessible via the `man` command. Shipped with software packages. Organized into sections.
    - **Man Page Sections**: Sections categorize content (e.g., 1 for user commands, 5 for file formats, 8 for administration). Section numbers are used in parentheses when referring to specific man pages (e.g., `passwd(1)`, `passwd(5)`).
    - **Man Page Structure**: Typically includes headings like NAME, SYNOPSIS, DESCRIPTION, OPTIONS, EXAMPLES, FILES, SEE ALSO. Optimized for printed output.
    - **Keyword Search**: Searching man pages by keywords in their titles and descriptions (`man -k`). Relies on a database updated by `mandb(8)`.
    - **Full-Text Search**: Searching for a keyword anywhere within the content of man pages (`man -K`).
- **GNU Info Documentation**: A different online documentation system developed by the GNU Project. Often more comprehensive than man pages for software packages as a whole. Structured as hypertext documents with hyperlinks, a browsable index, and full-text search. Viewed using the `pinfo` command. More flexible than man pages for describing complex concepts.
- **Documentation Comparison**: Man pages are typically single-topic references optimized for print, while Info documents are often multi-topic, hyperlinked guides optimized for browsing. Both systems are important resources.
- **Document Conventions**: Notes and Important boxes highlight useful tips and critical information.

Important Commands:

- **`man`**: Views manual pages.
    - `man topic`: Views the man page for a topic.

  - `man section topic`: Views the man page for a topic from a specific section.
  - `man -k keyword` (or `apropos keyword`): Searches man page titles and descriptions by keyword.
  - `man -K keyword`: Performs a full-text search of man pages.
  - `man -t topic`: Formats a man page for printing using PostScript. Output needs redirection (`>`) to a file.
- **`pinfo`**: Views GNU Info documentation.
  - `pinfo`: Starts at the top directory.
  - `pinfo topic`: Opens a specific Info topic.
  - Navigation within `pinfo` includes arrow keys, Enter to select, `U` to move up, `D` to go to the directory, `/` for search, `Q` to quit.
- **`mandb(8)`**: Updates the man page index used by `man -k`.
- **`evince(1)`**: A document viewer, can be used for PostScript files.
- **`lp(1)`**: Command to print files; the `-P` option can specify page ranges.

Applications:

- Finding information about specific commands, configuration files, system calls, and other system components.
- Consulting detailed documentation for software packages, especially those developed by the GNU Project.
- Troubleshooting problems by looking up relevant commands or configuration file formats.
- Searching for commands related to a specific task using keyword searches.
- Learning how to prepare man pages for printing or viewing formatted documentation.
- Navigating complex documentation effectively using hypertext links and search features in Info.

Chapter 5: Creating, Viewing, and Editing Text Files

Key Concepts:

- **Text Files**: Commonly used in Linux to store information and configuration settings. Can be viewed and edited with simple text editors.
- **Shell Redirection**: Changing where the standard output or standard error of a command is sent.
  - **Standard Output (`stdout`)**: The normal output of a command. Redirected using `>` (overwrite) or `>>` (append).
  - **Standard Error (`stderr`)**: Error messages from a command. Redirected using `2>` (overwrite) or `2>>` (append). Redirecting to `/dev/null` discards the output.
  - Redirecting both stdout and stderr can be done with `&>`.
- **Pipes (`|`)**: Connecting the standard output of one command to the standard input of another command. Allows processing data through a sequence of commands.

- **`tee` Command**: Used with pipes to read standard input and write it to both standard output and one or more files.
- **Vim Editor**: A powerful and widely used text editor available from the command line. Has different modes:
    - **Command Mode**: Default mode for navigation, deletion, copy/paste, and executing editor commands (e.g., `:wq`).
    - **Insert Mode**: Mode for typing text into the file. Entered by keys like `i`.
    - **Visual Mode**: Mode for selecting text. `v` for character selection, `V` for line selection, `Ctrl+v` for block selection.
- **`vimtutor`**: An interactive tutorial for learning Vim basics.
- **Shell Variables**: Variables defined within the shell environment.
- **Environment Variables**: Variables that are passed from the shell to programs executed by the shell. Can influence program behavior (e.g., `EDITOR` variable specifies the preferred text editor). Can be set in Bash startup scripts.

Important Commands:

- **`>`**: Redirects stdout, overwriting the target file.
- **`>>`**: Redirects stdout, appending to the target file.
- **`2>`**: Redirects stderr, overwriting the target file.
- **`&>`**: Redirects both stdout and stderr, overwriting the target file.
- `|`: Pipes stdout of one command to stdin of another.
- **`tee`**: Copies stdin to stdout and files.
- **`vim`**: Launches the Vim text editor.
- **`vimtutor`**: Starts the Vim interactive tutorial.
- **`export`**: Marks a shell variable for export to child processes, making it an environment variable.
- **`echo`**: Prints text to standard output.

Applications:

- Saving the output or error messages of commands to files for logging or further processing.
- Combining multiple commands to perform complex data manipulation using pipes.
- Editing system configuration files or scripts which are often text-based.
- Performing basic and advanced text manipulation using Vim's various modes and commands.
- Customizing the shell environment and program behavior using shell and environment variables.

Chapter 7: Controlling Access to Files

Key Concepts:

- **File Permissions**: Define the read, write, and execute access rights for files and directories.
- **Permission Categories**: Permissions are assigned based on three categories relative to the file/directory:
  - **User (Owner)**: The specific user who owns the file.
  - **Group (Group Owner)**: The group that owns the file.
  - **Others**: All other users on the system who are not the owner and not members of the group owner.
- **Permission Precedence**: The most specific permission category that applies to a user takes effect. User permissions override group permissions, which override other permissions.
- **Permission Representation**:
  - **Symbolic (Letters)**: Uses characters `r` (read), `w` (write), `x` (execute), `-` (no permission) for each category (user, group, others). A typical long listing (`ls -l`) shows permissions like `-rwxr-xr-x`.
  - **Numeric (Octal)**: Uses a three-digit octal number (0-7) for user, group, and others respectively. Read = 4, Write = 2, Execute = 1. Sum the values for each category.
- **Default Permissions**: Permissions automatically assigned to new files and directories when they are created. Can be influenced by the `umask` setting.
- **Special Permissions**: Additional permissions like SUID, SGID, and sticky bit that affect execution or file creation in specific ways.

Important Commands:

- **`ls -l`**: Displays detailed file information including permission string, owner, and group owner.
- **`chmod`**: Changes the permissions of files and directories. Can use symbolic (e.g., `u+x`, `g-w`, `o=r`) or numeric (e.g., `755`, `640`) notation.
- **`chown`**: Changes the user owner of a file or directory. Root privileges are typically required.
- **`chgrp`**: Changes the group owner of a file or directory.

Applications:

- Restricting or granting read, write, and execute access to files and directories for different users and groups.
- Implementing security policies by controlling who can access sensitive data or execute programs.
- Assigning ownership of files to specific users or groups after creation or transfer.
- Troubleshooting permission-related errors.
- Configuring directories where files created by different users should have specific group ownership or default permissions.
- Finding files based on their specific permission settings.

Chapter 8: Managing Processes

Key Concepts:

- **Process**: A running instance of an executable program. Each process has its own memory space, security credentials, execution threads, and state.
- **Process State**: The current condition of a process (e.g., running, sleeping, stopped (suspended), zombie (Z)).
- **Foreground vs. Background Processes**: Processes can run in the foreground, interacting with the terminal, or in the background, running independently.
- **Terminal Session**: Each terminal typically constitutes its own session with its own set of foreground and background jobs.
- **Signals**: Software interrupts used to report events to running programs. Programs can react to signals. Common signals include SIGINT (interrupt, typically Ctrl+C), SIGTSTP (stop, typically Ctrl+Z), SIGQUIT (quit, typically Ctrl+), and SIGKILL (forcible kill, cannot be ignored).
- **Load Average**: A metric provided by the Linux kernel that estimates the average number of processes that are either running or waiting to run over specific time periods (typically 1, 5, and 15 minutes). It indicates perceived system load.
- **Process Information**: Tools can display details about processes, including Process ID (PID), User, CPU usage, Memory usage, Start time, State, and Command.

Important Commands:

- **`ps`**: Lists processes. Options like `aux` or `ef` provide different levels of detail and filtering.
- **`top`**: Provides a dynamic real-time view of system processes, displaying CPU usage, memory usage, load average, and process details. Supports interactive filtering and sorting.
- **`uptime`**: Shows how long the system has been running and the load average.
- **`w`**: Shows who is logged in and what they are doing, including load average.
- **`jobs`**: Lists processes started in the background within the current terminal session.
- **`kill`**: Sends a signal to a process specified by its PID.
- **`pkill`**: Sends a signal to processes matching a name or other criteria.
- **`killall`**: Sends a signal to all processes matching a given name.

Applications:

- Identifying currently running programs on the system.
- Monitoring system performance and resource usage by observing CPU and memory consumed by processes.
- Diagnosing system slowdowns or unresponsiveness by checking the load average and identifying resource-intensive processes.
- Stopping, suspending, or terminating processes that are misbehaving or consuming too many resources.

- Managing background jobs initiated from a terminal.

Chapter 9: Controlling Services and Daemons

Key Concepts:

- **Services and Daemons**: Programs that run in the background, often started automatically at boot, to provide system functionalities or network services (e.g., web server, SSH server, logging service).
- **Systemd**: The primary system and service manager used in Red Hat Enterprise Linux 8. It is responsible for initializing the system and managing services.
- **Systemd Units**: Systemd manages various resources using unit files. Service units (`.service`) control daemons and services. Socket units (`.socket`) can be used to activate services on demand when network connections are received.
- **Service Control**: Systemd provides commands to manage the state (start, stop, restart, reload) and behavior (enable, disable) of services.
- **Service Status**: It's important to be able to check if a service is currently running or if it is configured to start automatically.

Important Commands:

- **`systemctl`**: The main command-line tool for controlling systemd. Requires root privileges (or `sudo`) for most actions.
  - **`systemctl status service_name`**: Displays detailed status information about a service, including whether it is active (running) and enabled (configured to start at boot).
  - **`systemctl start service_name`**: Starts a service.
  - **`systemctl stop service_name`**: Stops a service.
  - **`systemctl restart service_name`**: Stops and then starts a service.
  - **`systemctl reload service_name`**: Reloads the configuration file of a service without restarting the entire service, if supported.
  - **`systemctl enable service_name`**: Configures a service to start automatically during the boot process.
  - **`systemctl disable service_name`**: Prevents a service from starting automatically during the boot process.
  - **`systemctl is-active service_name`**: Checks if a service is currently running.
  - **`systemctl is-enabled service_name`**: Checks if a service is configured to start at boot.
  - **`systemctl list-units --type=service`**: Lists loaded and active service units.
  - **`systemctl list-unit-files --type=service`**: Lists all installed service unit files and their enabled/disabled status.

  - **`systemctl list-dependencies service_name`**: Displays the dependencies of a service.

Applications:

- Starting or stopping services for maintenance (e.g., updating configuration, applying patches).
- Ensuring necessary services are running on the system.
- Controlling which services start automatically when the system boots to optimize resources and security.
- Troubleshooting service-related issues by checking service status and logs.
- Configuring specific services like `rsyslog` for logging.

Chapter 10: Configuring and Securing SSH

Key Concepts:

- **SSH (Secure Shell)**: A cryptographic network protocol for secure communication, particularly for accessing remote command lines. Encrypts data transmitted over the network.
- **OpenSSH**: A popular suite of applications implementing the SSH protocol. The server component is the `sshd` daemon.
- **`sshd_config`**: The main configuration file for the OpenSSH server, located at `/etc/ssh/sshd_config`. Allows customizing server behavior.
- **Authentication Methods**: SSH supports different ways to verify user identity:
  - **Password Authentication**: Users log in using a username and password. Can be disabled for enhanced security.
  - **Key-based Authentication**: Uses a pair of cryptographic keys (a private key and a public key) to authenticate without a password. More secure than passwords for automated tasks and often preferred for interactive logins.
    - **Private Key**: Must be kept secret by the user.
    - **Public Key**: Can be shared and is placed on the remote server in the user's `~/.ssh/authorized_keys` file.
- **Host Keys**: SSH servers have host keys to identify themselves to clients. Clients store these identities in `~/.ssh/known_hosts` or `/etc/ssh/ssh_known_hosts` to detect potential man-in-the-middle attacks.
- **Security Best Practices**: Common security enhancements for SSH include disabling direct root login and disabling password-based authentication.

Important Commands:

- **`ssh hostname`** or **`ssh user@hostname`**: Connects to a remote system using SSH.
- **`ssh-keygen`**: Generates a public/private SSH key pair.

- **`ssh-copy-id user@hostname`**: Copies a user's public key to the `authorized_keys` file on a remote server, enabling key-based authentication.
- **Editing `/etc/ssh/sshd_config`**: Requires a text editor and root privileges (e.g., `sudo vim /etc/ssh/sshd_config`). Parameters like `PermitRootLogin no` and `PasswordAuthentication no` can be set.
- **`systemctl restart sshd`**: Restarts the SSH daemon after making configuration changes to `/etc/ssh/sshd_config`.

Applications:

- Accessing the command line of remote servers securely.
- Transferring files securely between systems using tools like `scp`, `sftp`, or `rsync` over SSH.
- Enabling passwordless logins for users or automated scripts using key-based authentication.
- Enhancing the security posture of a server by restricting direct root access via SSH and requiring key-based authentication instead of passwords.
- Managing remote systems from a central administration host.

Chapter 11: Analyzing and Storing Logs

Key Concepts:

- **System Logging**: The process by which the operating system and applications record events, errors, and other messages. Logs are crucial for auditing, monitoring, and troubleshooting.
- **Syslog Protocol**: A standard protocol used by many programs to send log messages. Messages are tagged with a **facility** (source/type of message, e.g., mail, kernel, daemon) and a **priority** (severity level, e.g., debug, info, warning, error).
- **Logging Daemons**: Services responsible for receiving, processing, and storing log messages.
  - **`rsyslogd`**: A logging daemon that processes syslog messages and typically writes them to text files. Configuration is in `/etc/rsyslog.conf` and files in `/etc/rsyslog.d/`.
  - **`systemd-journald`**: The daemon managing the systemd journal. Captures logs from various sources (kernel, systemd units, syslog).
- **Log Storage**:
  - **Text Files**: Traditionally stored in the `/var/log` directory. Can be viewed with tools like `less` or `tail`.
  - **System Journal**: Structured, indexed storage for log data. By default, stored temporarily in `/run/log/journal` and cleared on reboot. Can be configured to persist across reboots by changing the `Storage` setting in `/etc/systemd/journald.conf` to `persistent`.

- **Journal Fields**: Journal entries have fields like `_COMM` (command name), `_EXE` (executable path), `_PID` (process ID), `_UID` (user ID), `_SYSTEMD_UNIT` (service unit) which can be used for searching.
- **Log Rotation**: A process to archive or delete old log files to prevent them from consuming excessive disk space.
- **Accurate Time**: Maintaining synchronized system time is vital for correlating log entries across multiple systems. **NTP (Network Time Protocol)** is used for time synchronization. The `chronyd` service implements NTP. Time zones must be correctly configured.

Important Commands:

- **`journalctl`**: Views the systemd journal.
  - `journalctl`: Displays all journal entries.
  - `journalctl -f`: Follows the end of the journal (like `tail -f`).
  - `journalctl _FIELD=value ...`: Filters journal entries based on fields (e.g., `journalctl _SYSTEMD_UNIT=sshd.service`). Multiple filters can be combined.
  - `journalctl -u unit_name`: Views logs specifically for a systemd unit.
  - `journalctl --since "YYYY-MM-DD HH:MM:SS"` / `--until "..."`: Views logs within a time range.
  - `journalctl -p priority`: Filters by priority level (e.g., `err`, `warning`, `info`).
- **`tail /var/log/somefile.log`**: Views the end of a traditional log file.
- **`less /var/log/somefile.log`**: Views a traditional log file one screen at a time.
- **`timedatectl`**: Views and changes system time, date, and time zone settings, and enables/disables NTP synchronization.
  - `timedatectl`: Shows current time, time zone, and NTP status.
  - `timedatectl list-timezones`: Lists available time zones.
  - `timedatectl set-timezone "TimeZone"`: Sets the system time zone.
  - `timedatectl set-ntp true/false`: Enables/disables NTP synchronization.
- **`chronyc`**: Command-line interface to manage the `chronyd` service. (e.g., `chronyc sources`)
- **Editing `/etc/rsyslog.conf` or files in `/etc/rsyslog.d/`**: Requires a text editor and root privileges (e.g., `sudo vim /etc/rsyslog.d/grading-debug.conf`) to configure rsyslog rules.
- **`systemctl restart rsyslog`**: Restarts the rsyslog service after configuration changes.
- **`systemctl status systemd-journald`**: Checks the status of the journal service.
- **Editing `/etc/systemd/journald.conf`**: Requires a text editor and root privileges to configure journal persistence.

Applications:

- Troubleshooting system issues by examining error messages and event sequences recorded in logs.
- Auditing system activity by reviewing log entries related to user logins, service status changes, etc..
- Creating custom log files for specific types of messages or applications using rsyslog configuration.
- Searching the system journal efficiently for specific events based on process, user, service, or time.
- Ensuring logs are preserved across system reboots by configuring journal persistence.
- Maintaining accurate time stamps in logs by synchronizing system time using NTP and setting the correct time zone.

Chapter 12: Managing Networking

Key Concepts:

- **Networking Concepts**: Fundamental principles including network addressing (IP addresses, netmasks), routing (default gateway), and name resolution (DNS).
- **Network Configuration**: Setting up network interfaces with IP addresses, netmasks, gateways, and DNS servers. Can be static (manually assigned) or dynamic (via DHCP).
- **NetworkManager**: A dynamic network service in RHEL that manages network connections and configurations. It provides tools to configure networking from the command line or GUI.
- **Connection Profiles**: Sets of network configurations (IP address, gateway, DNS, etc.) associated with a specific network interface.
- **Validating Configuration**: Checking the current network settings applied to interfaces.
- **Network Connectivity Testing**: Verifying communication with other hosts on the network or the internet.
- **Host Names**: Names assigned to systems for easier identification than IP addresses.
    - **Static Host Name**: Stored in the `/etc/hostname` file and persists across reboots.
    - **Transient Host Name**: Set by DHCP or other services and is not persistent.
    - **Pretty Host Name**: A user-friendly descriptive name.
- **Name Resolution**: The process of translating host names into IP addresses (and vice versa). Often handled by DNS (Domain Name System) servers. Local resolution can use the `/etc/hosts` file.

Important Commands:

- **`nmcli`**: Command-line tool for controlling NetworkManager.
    - `nmcli device status`: Lists network interfaces and their state.
    - `nmcli connection show`: Lists active and inactive network connections/profiles.

    - `nmcli connection add type ethernet con-name ProfileName ifname InterfaceName ...`: Creates a new connection profile (e.g., setting IP, gateway, DNS).
    - `nmcli connection modify ProfileName ...`: Modifies an existing connection profile.
    - `nmcli connection up ProfileName`: Activates a connection profile.
    - `nmcli connection down ProfileName`: Deactivates a connection profile.
- **`ip`**: Tool to show and configure network devices and routing.
    - `ip addr show` or `ip a`: Displays IP addresses and interface configuration.
    - `ip route show` or `ip r`: Displays the routing table.
- **`ping destination`**: Tests network connectivity to a host by sending ICMP packets.
- **`ss`**: Utility to investigate sockets, showing listening ports and network connections. (e.g., `ss -tulnp` shows listening TCP and UDP ports with process info).
- **`hostnamectl`**: Views and changes the system's static, transient, and pretty host names.
    - `hostnamectl`: Shows current host name status.
    - `hostnamectl set-hostname StaticHostname`: Sets the static host name.
- **`hostname`**: Displays or temporarily sets the system's host name.
- **`cat /etc/hostname`**: Displays the content of the static host name file.

Applications:

- Configuring network interfaces with static IP addresses or enabling DHCP.
- Setting the default gateway to allow communication outside the local network.
- Configuring DNS servers for name resolution.
- Verifying that network settings are correctly applied and that connectivity exists to other systems or the internet.
- Troubleshooting network problems by inspecting interface status, routing tables, and open ports.
- Setting and managing the system's identity on the network using host names.
- Ensuring systems can resolve host names to IP addresses using DNS or local files.

Chapter 13: Archiving and Transferring Files

Key Concepts:

- **Archiving**: Combining multiple files and directories into a single file (an archive) for easier storage, backup, or transfer.
- **Compression**: Reducing the size of files, often applied to archives to save disk space and transfer time. Gzip is a common compression method.
- **File Transfer**: Copying files between different locations, including across a network between different systems.

- **Secure Transfer**: Using encrypted protocols to transfer files across a network, protecting data confidentiality and integrity. SSH is commonly used as the underlying secure channel.
- **Synchronization**: Copying files efficiently by only transferring the parts that have changed since the last copy, minimizing data transfer.

Important Commands:

- **`tar`**: Command-line utility for creating, viewing, and extracting tar archives. Can also handle compression.
    - `tar -cf archive.tar file1 dir2`: Creates a tar archive (`-c`) named `archive.tar` (`-f`) containing `file1` and `dir2`.
    - `tar -czf archive.tar.gz file1 dir2`: Creates a gzipped tar archive (`-z`).
    - `tar -xf archive.tar`: Extracts (`-x`) contents from a tar archive (`-f`).
    - `tar -xzf archive.tar.gz`: Extracts contents from a gzipped tar archive.
    - `tar -tvf archive.tar`: Lists (`-t`) contents verbosely (`-v`) from a tar archive (`-f`).
- **`gzip`**: A compression utility. Often used with `tar`.
- **`scp`**: Secure Copy Protocol, a command-line utility for copying files and directories over an SSH connection.
    - `scp /local/path user@remote:/remote/path`: Copies from local to remote.
    - `scp user@remote:/remote/path /local/path`: Copies from remote to local.
    - `scp -r /local/dir user@remote:/remote/dir`: Copies directories recursively (`-r`).
    - `scp -i /path/to/private_key file user@remote:/path`: Uses a specific identity file for authentication.
- **`sftp`**: Secure File Transfer Protocol, an interactive command-line program for file transfer over SSH. Provides commands like `get`, `put`, `ls`, `cd` within an sftp session.
- **`rsync`**: A command-line utility for synchronizing files and directories. Uses an algorithm to transfer only the differences, making it efficient for updates. Can use SSH as a transport for secure synchronization.
    - `rsync -avz /source/path/ user@remote:/destination/path/`: Synchronizes recursively (`-r` or implied by `-a`), verbosely (`-v`), compressing data (`-z`), preserving attributes (`-a`).

Applications:

- Creating backups of important files and directories.
- Packaging multiple files into a single archive for easier distribution or storage.

- Restoring files from backups.
- Securely transferring files between servers or between a workstation and a server.
- Performing efficient, incremental backups or file synchronization between systems using `rsync`.

Chapter 14: Installing and Updating Software Packages

Key Concepts:

- **Software Packages**: Software components in RHEL are distributed as RPM (Red Hat Package Manager) packages. Packages contain the necessary files, metadata, and instructions for installing, upgrading, or uninstalling software.
- **RPM**: The low-level package format and command-line tool. Can install individual package files and query installed packages. Does not automatically handle dependencies or work with online repositories efficiently.
- **Yum (Yellowdog Updater, Modified)**: A higher-level command-line tool built on top of RPM (in RHEL 8, this is often implemented by `dnf`). Designed to work with software repositories and automatically resolve package dependencies.
- **Software Repositories**: Centralized locations on the network or locally where RPM packages are stored. Configured in files in `/etc/yum.repos.d/`. Can be enabled or disabled.
- **Dependencies**: Most software packages rely on other packages to function correctly. Package managers like Yum/DNF automatically identify and install these dependencies.
- **Red Hat Subscription Management**: Tools (like `subscription-manager`) used to register RHEL systems and assign entitlements (subscriptions) to access official Red Hat software repositories and support resources.
- **Application Streams**: A feature in RHEL 8 allowing multiple versions of applications and their dependencies to be available in the standard repositories simultaneously.
- **Modules**: A mechanism within Application Streams to manage the installation of specific versions (streams) of software components. Modules have streams that can be listed, enabled, and switched.

Important Commands:

- **`rpm`**: Low-level package management tool.
  - `rpm -qa`: Lists all installed RPM packages.
  - `rpm -qi package_name`: Displays information about an installed package.
  - `rpm -ql package_name`: Lists files installed by a package.
  - `rpm -ivh package.rpm`: Installs a package file (`-i`), verbosely (`-v`), showing hash marks (`-h`).
- **`yum`** (or **`dnf`**): Higher-level package management tool.
  - `yum search keyword`: Searches repositories for packages matching a keyword.

- `yum info package_name`: Displays information about a package from repositories.
- `yum install package_name`: Downloads and installs a package and its dependencies from configured repositories.
- `yum update` or `yum update package_name`: Updates all packages or a specific package.
- `yum remove package_name`: Removes a package.
- `yum repolist`: Lists configured repositories.
- `yum history`: Views the transaction history of yum/dnf operations.
- `yum module list`: Lists available package modules and streams.
- `yum module enable module_name:stream`: Enables a specific module stream.
- `yum module switch module_name:stream`: Switches to a different module stream.
- `yum module install module_name:stream`: Installs packages from a module stream.

- **`subscription-manager`**: Tool for registering and managing subscriptions.
  - `subscription-manager register --auto-attach`: Registers the system and attaches available subscriptions.
  - `subscription-manager list --consumed`: Lists consumed subscriptions.
  - `subscription-manager repolist`: Lists enabled repositories based on subscriptions.

Applications:

- Installing new software packages required for specific tasks or services.
- Keeping the system's software up-to-date with security patches and bug fixes.
- Removing unnecessary software.
- Managing software dependencies automatically.
- Accessing official Red Hat software updates and support by registering the system.
- Installing specific versions of software using Application Streams and modules.
- Controlling which software sources (repositories) the system uses.

Chapter 15: Accessing Linux File Systems

Key Concepts:

- **File Systems**: The method used to organize and store files on a storage device. Linux supports various file system types (e.g., XFS, ext4). File systems are mounted into the overall file-system hierarchy.
- **Block Devices**: Storage devices (like hard drives, SSDs, partitions) that transfer data in fixed-size blocks. In Linux, block devices are represented by special files, typically found in the `/dev/` directory (e.g., `/dev/vda`, `/dev/vda1`, `/dev/sdb`).

- **Mount Point**: A directory in the file-system hierarchy where a file system from a block device is attached to make its contents accessible.
- **Mounting**: The process of making the files and directories on a file system available for access by attaching it to a mount point. File systems need to be mounted before their data can be read or written. Manual mounting usually requires root privileges.
- **Unmounting**: The process of detaching a file system from its mount point, making its contents inaccessible through that point. Necessary before removing a storage device.
- **File Location**: Finding specific files or types of files within the file system hierarchy.
- **File Searching Tools**: Utilities designed to locate files based on criteria like name, type, size, permissions, owner, etc..

Important Commands:

- **`lsblk`**: Lists block devices in a tree format, showing their relationships (disks, partitions, mount points). Useful for identifying available storage devices.
- **`blkid`**: Locates and prints block device attributes, including UUID, file system type, and partition label.
- **`df -h`**: Displays disk space usage for mounted file systems in a human-readable format.
- **`mount`**: Mounts a file system.
  - `mount device mount_point`: Mounts the file system on `device` at `mount_point`.
  - `mount -a`: Mounts all file systems listed in `/etc/fstab`.
- **`umount`**: Unmounts a file system.
  - `umount mount_point`: Unmounts the file system mounted at `mount_point`.
  - `umount device`: Unmounts the file system on `device`.
- **`find path criteria action`**: Searches for files and directories starting from `path` and performs an `action` on those matching `criteria`. Searches the file system in real-time.
  - `-name pattern`: Finds files by name (case-sensitive).
  - `-iname pattern`: Finds files by name (case-insensitive).
  - `-type c`: Finds objects of a specific type (`f`=file, `d`=directory, `l`=symbolic link, `b`=block device).
  - `-user username`: Finds files owned by `username`.
  - `-group groupname`: Finds files owned by `groupname`.
  - `-perm permissions`: Finds files with specific permissions (e.g., `-perm 640`).
  - `-size [+|-]N[cwbkMG]`: Finds files by size. `+N` for greater than N, `-N` for less than N, `N` for exactly N. Suffixes specify units (c=bytes, k=KB, M=MB, G=GB).
  - `-print`: Prints the full file name (default action if none specified).
  - `-exec command {} \;`: Executes `command` on found files.
  - Errors can be redirected to `/dev/null` (e.g., `find ... 2>/dev/null`) to hide permission errors in search results.

- **`locate keyword`**: Searches a pre-built database for file names or paths matching `keyword`. Much faster than `find` but relies on the database being up-to-date. The database is updated by the `updatedb` command (often run automatically).

Applications:

- Accessing data stored on hard drives, USB drives, optical media, or network shares by mounting their file systems.
- Determining where specific directories or files are stored (which block device).
- Creating and managing mount points for new or existing file systems.
- Searching the file system to find files based on various criteria like name, size, owner, or permissions.
- Finding configuration files, executable programs, or user data.

Chapter 16: Accessing Red Hat Support

Key Concepts:

- **Red Hat Customer Portal (access.redhat.com)**: A central online resource for Red Hat customers. Provides access to documentation, software downloads, technical expertise, and support case management.
- **Knowledgebase**: A searchable database on the Customer Portal containing solutions, FAQs, technical articles, and troubleshooting guides.
- **Red Hat Insights**: A Software-as-a-Service (SaaS) tool that provides predictive analytics for RHEL systems registered with Red Hat. Analyzes system configurations to identify potential security risks, performance bottlenecks, availability issues, and stability threats. Registered hosts are listed in the Inventory tab on the Customer Portal. The `insights-client` command is used to register systems.
- **Support Cases**: Customers can open and manage technical support requests through the Customer Portal or using command-line tools.
- **Red Hat Support Tool**: A command-line tool (`redhat-support-tool`) for interacting with Red Hat Support, including viewing and modifying cases. Can also perform diagnostics like extracting backtraces from kernel crash dumps created by `kdump`.

Important Commands:

- **`redhat-support-tool`**: Command for interacting with Red Hat Support cases and performing diagnostics. (e.g., `redhat-support-tool modifycase --status=Closed 01034421`).
- **`insights-client --register`**: Registers a system with Red Hat Insights.
- **`kdump`**: A service/command used to capture a kernel crash dump file when the system crashes, useful for diagnosing kernel issues.

Applications:

- Finding solutions to technical problems and answering questions by searching the Red Hat Knowledgebase.
- Identifying potential issues on RHEL systems before they cause problems by using Red Hat Insights analysis.
- Opening and managing technical support cases with Red Hat.
- Using specialized tools like the Red Hat Support Tool for advanced diagnostics or interacting with support cases from the command line.
- Registering RHEL systems to gain access to entitled repositories, updates, and support services.