Download as PDF

# Red Hat System Administration I (RHSA1) Study Guide Summary

## I. Day 1: Fundamentals, Linux Components, and File System Basics

### 1. Free/Open Source Software (FOSS) and Licensing

- **Definition:** FOSS provides users with significant freedoms.
- **Freedoms Provided by FOSS:**
  - The ability to View the source code used to compile programs.
  - The ability to Make modifications to the software.
  - The ability to Distribute these modifications.
- **FOSS Licenses:** FOSS is usually covered under a public license. An open-source license allows the source code, blueprint, or design to be used, modified, and/or shared under defined terms and conditions.
- **Key Examples:** The most common public license is the GNU General Public License (GPL). Other examples include LGPL, Apache, Mozilla Public License, and BSD.

### 2. Linux History and Relevance

- **Unix Origin:** The first version of Unix was created in Bell Labs in 1969.
- **Unix Flavors:** Historical Unix variants include AIX (IBM), HP/UX (Hewlett-Packard), Solaris (Sun), and IRIX (Silicon Graphics). These variants generally operate in the same manner and offer the same standard utilities and commands.
- **Linux Kernel:** The Linux kernel was created by Linus Torvalds after he finished college in 1991.
- **Why Linux?** Linux is dominant in the professional and servers sector. Internet

service providers (ISPs), e-commerce sites, and commercial applications widely use Linux.

- **Why Red Hat Enterprise Linux (RHEL)?**
  - RHEL is reliable, secure, and scales well.
  - More than 90% of Fortune Global 500 companies utilize Red Hat products and solutions.
  - Red Hat offers training and support and partners with hardware vendors.

### 3. Linux Components

Linux is composed of three primary parts:

1. **Kernel:**
   - The core of the operating system.
   - Contains components like device drivers.
   - Loads into Random Access Memory (RAM) when the machine boots and remains resident until the machine powers off.
2. **Shell:**
   - Provides an interface allowing the user to communicate with the kernel.
   - Parses commands entered by the user and translates them into logical segments for execution by the kernel or other utilities.
   - "bash" (Bourne Again Shell) is the most commonly used shell on Linux. Other shells include Bourne Shell (sh), Korn Shell (ksh), and C Shell (csh).
3. **Terminal:**
   - The display area that provides the shell with a place to accept typed commands and display their results.

### 4. Command Structure and Execution

- **Command Syntax:** `command [options] [arguments]`
  - Each item in the syntax is separated by a space.
  - Options: Modify the command's behavior.
  - Arguments: Information needed by the command, often file names.
- **Multiple Commands:** Separate commands with a semicolon (;).

- **Interrupting/Terminating:**
  - Interrupt: Use [Ctrl]-c to interrupt a command that is taking too long to execute.
  - Terminate Input: Use [Ctrl]-d to terminate a command that expects input to come from the keyboard but hasn't received an argument.

### 5. Linux Documentation

Linux documentation is often accessed via manual pages (man).

- **Manual Page Sections:** A manual page typically consists of:
  - Name: The command name and a one-line description.
  - Synopsis: The formal syntax of the command.
  - Description: Explanation of how the command works.
  - Files: The files used by the command.
  - Bugs: Known errors.
  - See also: Other related commands.
- **Documentation Commands:**
  - `man -k keyword` : Shows commands that have manual pages containing the given keyword.
  - `whatis command` : Shows the command's one-line description.
  - `--help Option` : A way to get help about a command, often built into the command itself (if supported).

### 6. File and Directory Management

- **Directory Structure Concept:** Analogous to a building (file system), a room (directory), and a desk (file).
- **Key Concepts:**
  - Current Working Directory (CWD): The directory (room) you are currently in.
  - Pathnames: Can be absolute (full path starting from root) or relative (relative to the CWD).
- **Directory Commands:**

    - `pwd` : Prints the current working directory (e.g., /home/guest).
    - `cd /path/` : Changes directory.
    - `cd ..` : Moves to the parent directory.
    - `cd ~` : Moves to the home directory.
    - `cd -` : Moves to the previous working directory.
- **Listing Directory Contents (ls):**
    - `ls -a` : Lists all files, including hidden files (those starting with a dot .).
    - `ls -l` : Provides a long listing, showing detailed information (permissions, owner, size, date).
    - `ls -F` : Appends indicators to filenames (e.g., / for directories, * for executables, @ for symbolic links).
    - `ls -R` : Recursively lists subdirectories.
    - `ls -ld dir1` : Shows the long listing of the directory itself.
- **File Naming Rules:**
    - File names can be up to 255 characters.
    - File names are case sensitive.
    - Important Note: While conventions exist, there are no extensions required in Linux.
    - Avoid: Special characters such as >< ? * # '.
- **Viewing File Content:**
    - `cat fname` : Displays the file content.
    - `more fname` : Displays content one screen at a time.
        - more Scrolling Keys: Spacebar (forward one screen), Return (scroll one line), b (move back one screen), /string (search forward for pattern), q (quit).
    - `head -n fname` : Displays the beginning lines of a file.
    - `tail [-n|+n] fname` : Displays the ending lines of a file.

### 7. File Globbing (Metacharacters)

Wildcards, or "metacharacters," allow one pattern to expand to multiple filenames when executing commands.

| Metacharacter | Definition | Example Application |
|---|---|---|
| * (Asterisk) | Represents 0 or more characters (except the leading dot .). | ls f* matches file1, file2, fruit. |
| ? (Question Mark) | Represents any single character (except the leading dot .). | ls file? matches file1, file2, file4. |
| [] (Square Bracket) | Represents a range of characters for a single character position. | ls [a-f]* matches files starting with a, b, c, d, e, or f. |

### 8. File and Directory Manipulation Commands

- **Copying:** `cp options source(s) target`
  - `-i` : Prevents accidental overwriting of existing files.
  - `-r` : Copies a directory, including the contents of all subdirectories (recursive).
- **Moving and Renaming:** `mv options source(s) target`
  - `-i` : Prevents accidental overwriting of existing files/directories.
- **Creation:**
  - `touch file(s)_name` : Creates new, empty files.
  - `mkdir [-p] dir(s)_name` : Creates directories. The -p option creates parent directories as needed.
- **Removal:**
  - `rm [-i] file(s)_name` : Removes files. -i prompts before removal.
  - `rmdir dir(s)_name` : Removes directories (only if empty).
  - `rm [-r] dir(s)_name` : Recursively removes directories and their contents.

## II. Day 2: User/Group Administration and Permissions

### 1. User and Group Account Files

Linux systems rely on several key files to manage user and group identity and authentication:

| File Name | Purpose and Format | Key Fields Included |
|---|---|---|
| /etc/passwd | Defines user accounts and properties. | Login name, User ID (uid), Group Id (gid), Home Directory, Login shell. |
| /etc/shadow | Stores secure, encrypted password information and aging policy. | Login name, Encrypted password, Days since 1/1/1970 password last changed, Min/Max days between changes, Warning days before expiration, Expiration date. |
| /etc/group | Defines groups and their members. | Group name, Group ID (gid), Comma-separated list of group members. |

### 2. User Account Management Commands

- **Adding Users:**
  - `# useradd username` : Creates the user account; populates the user's home directory from the /etc/skel directory.
  - `# useradd -D` : Views and modifies default user addition settings.
  - `# passwd username` : Sets the user's password.
  - `# newusers filename` : Adds multiple user accounts listed in a file.
- **Modifying Users:**
  - `# usermod [options] username` : Command-line tool to change user account information.
    - Options: `-l <login name>` (changes login name), `-L` (lock password), `-U` (unlock password).
  - `chage` command: Used to set password aging policies.
- **Deleting Users:**
  - `# userdel [-r] username` : Deletes the user account. The -r option removes the user's home directory (/home/username) and mail spool file (/var/spool/mail/username).

### 3. Password Aging Policies (chage)

The chage command manages password expiration and warning settings.

- **Syntax:** `# chage [options] username` .
- **Options:**
  - `-m` : Change the minimum number of days between password changes.
  - `-M` : Change the maximum number of days between password changes.
  - `-E date` : Change the expiration date for the account.
  - `-W` : Change the number of days to start warning the user before a required password change.

### 4. Group Management Commands

- **Creation:** `# groupadd groupname` .
- **Modification:** `# groupmod [options] groupname` .
- **Deletion:** `# groupdel groupname` .
- **Advanced Group Management:** The gpasswd command can be used to define group members, group administrators, and create or change group passwords.
- **Finding Files:** `# find / -nogroup` : Lists all files owned by groups that are not defined in the /etc/group file.

### 5. Switching Accounts and Status Checks

- **Switching User Identity:** `# su [-] [username]` : Switches to another user account.
- **Switching Active Group:** `newgrp group` : Switches between groups the user is a member of.
- **Status Commands:**
  - `groups` : Displays the groups the user is a member of.
  - `whoami` : Displays the current effective user name and ID.
  - `id` : Displays the user's UID, GID, and groups.
  - `who` : Displays user login name, login device (tty), and login date/time for users currently on the system.
  - `w [user]` : Displays a summary of current system activity, including what each user is doing.
  - `finger` : Reveals detailed information about users, locally or remotely.
- **sudo Access:** sudo is a more secure way to grant limited root privileges. Access is

controlled by the /etc/sudoers file, which must be edited using the visudo editor (which includes a syntax checker).

### 6. Ownership and Permissions

- **Ownership:** Every file and directory has a user owner and a group owner. A newly created file is owned by the user who creates it and that user's primary group.
- **Changing Ownership:** `# chown user1 file1` , `# chown user1:group1 file1` , `# chown :group1 file1` .
- **Security Scheme:** Permissions protect files and directories and are assigned to three categories:
    1. Owner (u): File owner permissions.
    2. Group (g): Members of the group the file is assigned to.
    3. Other (o): All other users. Note: The most specific permissions apply. Only the file owner or root can change permissions.

| Permission Type | Access for a File | Access for a Directory |
|---|---|---|
| Read (r) | Display contents, copy the file. | List directory contents using ls. |
| Write (w) | Modify file contents. | Add and delete files (requires execute access too). |
| Execute (x) | Execute the file (if it's executable/script). | Use the cd command to access the directory. |

- **Changing Permissions (chmod):** `chmod permission filename` .
    - **Symbolic Mode:** Uses letters and operators.
        - Who: u, g, o, a (all).
        - Operator: + (add), - (remove), = (assign absolutely).
        - Permissions: r, w, x.
        - Example: `chmod u+x,go+r file1` .
    - **Octal (Numeric) Mode:** Uses numerical values.
        - Read (r): 4.

    - Write (w): 2.
    - Execute (x): 1.
    - Example: `chmod 755 file1` (Owner: 7=rwx; Group: 5=rx; Other: 5=rx).

- **Default Permissions (umask):** Sets the default permissions applied when new files and directories are created. Example: `# umask 002` .

#### 7. System Shutdown

Shutdown or reboot is only required when adding/removing hardware, or upgrading the OS/kernel.

- **Shutdown Commands:**
    - `shutdown -k now` : Sends a warning message and disables logins, but does not actually halt the system.
    - `shutdown -h time` : Halts the system after shutdown.
    - `poweroff` .
    - `init 0` .

## III. Day 3: The Vi Editor, Initialization, and Environment

#### 1. The Vi (Vim) Text Editor

Vi is the default editor in all UNIX operating systems and is available even in emergencies. Linux typically uses Vim (vi improved), which offers syntax highlighting, better arrow key support, and mouse support.

- **Vi Operating Modes:**
    1. Command Mode: The default mode upon startup. Used for issuing commands (delete, copy, movement).
    2. Edit Mode (Insert Mode): Used for entering text into the file. Accessed via commands like i, a, o, etc. Exit via Esc key.
    3. Last Line Mode: Used for advanced editing commands (saving, quitting, search/replace). Accessed by entering a colon (:) while in Command Mode.
- **Key Insertion Commands (from Command Mode):**
    - `i` : Inserts text before the cursor.

    - `a` : Appends text after the cursor.
    - `I` : Inserts text at the beginning of the line.
    - `A` : Appends text at the end of the line.
    - `o` : Opens a new blank line below the cursor.
    - `O` : Opens a new blank line above the cursor.
- **Key Editing and Deletion Commands (Command Mode):**
    - `x` : Deletes the character at the cursor.
    - `dw` : Deletes a word or part of the word to the right of the cursor.
    - `dd` : Deletes the entire line containing the cursor.
    - `D` : Deletes from the cursor to the right end of the line.
- **Cursor Movement (Command Mode):**
    - `h, j, k, l` : Standard movement keys (left, down, up, right).
    - `w` : Forward one word.
    - `0` : To the beginning of the line.
    - `G` : Goes to the last line of the file.
    - `nG` or `:n` : Goes to Line n.
    - Control-F / Control-B: Pages forward/back one screen.
- **Search and Replace:**
    - `/string` : Searches forward for the string.
    - `?string` : Searches backward for the string.
    - `n` : Searches for the next occurrence.
    - `:%s/old/new/g` : Searches for the old string and replaces it with the new string globally (Last Line mode).
- **Save and Quit (Last Line Mode):**
    - `:w` : Saves the file.
    - `:wq` , `:x` , or `ZZ` : Saves and quits.
    - `:q!` : Quits without saving (force quit).
- **Customizing Vi (Last Line Mode):**
    - `:set nu` : Shows line numbers.

- `:set ic` : Ignore case sensitive searches.

### 2. Shell Initialization Files

These files set up the user's environment upon login or shell invocation.

| File Type | File Name | Execution Context |
|---|---|---|
| Global Login | /etc/profile | Executed by the bash login shell and DisplayManager (desktop loads). |
| Global Shell | /etc/bash.bashrc | Executed whenever a user enters a shell or the desktop environment. |
| User Graphical/ Text | ~/.profile | Executed by DisplayManager (GUI session) and login shell (textual console). |
| User Login | ~/.bash_profile or ~/.bash_login | Executed by bash when started as a login shell (prefers ~/.bash_profile). Does not influence graphical sessions by default. |
| User Invocation | ~/.bashrc | Executed in every invocation of bash, including graphical login environment. |

### 3. Environment Variables

Variables that store data related to the current shell environment. Their value is accessed by preceding the name with a dollar sign ($). The set command views the contents of all variables.

| Variable Name | Meaning |
|---|---|
| $HOME | Complete path of the user's home directory. |
| $PATH | Colon-separated list of directories the shell uses to look for executable programs. |
| $PWD | The user's current working directory. |
| $USER | Currently logged in user. |

| | |
|---|---|
| $SHELL | Path name of the login shell. |
| $HOSTNAME | Name of the computer. |

### 4. Command Alias and History

- **Alias:** Used to create command shortcuts (e.g., `alias ll='ls -l '`).
  - To view all set aliases, type `alias`.
  - To remove an alias, use `unalias command`.
  - To bypass an alias, use the full path or a backslash (e.g., `\ls`).
- **Command History:** bash stores history in ~/.bash_history by default.
  - `!!` : Repeats the last command.
  - `!string` : Repeats the last command that started with string.
  - `!n` : Repeats a command by its number in history.
  - `!-n` : Repeats a command entered n commands back.
  - `^old^new` : Repeats the last command with a substitution (old replaced by new).

## IV. Day 4: Processes, I/O Redirection, and Utilities

### 1. Processes and Priority

- **Process Definition:** Every running program creates a process. Daemons are processes that run in the background and provide services. Every process has a PID (Process ID).
- **Parent/Child Relationship:** When a process creates another, the creator is the parent process, and the new one is the child process. The parent waits for the child to finish.
- **Scheduling Priority (Niceness):** Linux divides CPU time into slices. Niceness values affect scheduling priority.
  - Range: Niceness values range from -20 to +19.
  - Meaning: Smaller numbers represent higher priority. Default value is 0.
  - User Limits: Standard users can only adjust the priority down (up to +19, lower priority). Only the root user can increase priority (down to -20).

- **Adjusting Priority:**
  - At Invocation: `nice [-n adjustment] command` .
  - Running Process: `renice priority [[-p] pid ...]` .

### 2. Viewing and Controlling Processes

- **ps command:** Displays process status.
  - Output fields include PID, TTY (terminal identifier), Execution time, and Command name.
  - Options: `-e` (all system processes), `-f` (full information), `-u uid` (processes for a specific user).
- **top utility:** Views processes dynamically.
- **pgrep command:** Searches for processes by pattern. Options: `-x` (exact match), `-u uid` (specific user), `-l` (display name with PID).
- **Signals and Killing:** A signal is a message sent to a process to perform an action (e.g., termination).
  - SIGTERM (15): Default termination signal.
  - SIGKILL (9): Immediate termination signal (cannot be ignored).
  - `kill` : `kill [-signal] PIDs` .
  - `pkill` : `pkill [-signal] process_name` (kills based on process name).

### 3. Standard I/O and Redirection

- **Standard Input (0, stdin):** Data source for a command (typically keyboard).
- **Standard Output (1, stdout):** Data destination for command results (typically screen).
- **Standard Error (2, stderr):** Data destination for command errors and messages (typically screen).
- **Redirection Operators:**
  - `>` : Redirects standard output to a file (overwrites file contents).
  - `>>` : Redirects standard output to a file (appends to file contents).
  - `<` : Redirects file content as standard input to a command.
  - `2>` : Redirects standard error to a file (must place 2 before the operator).
  - `2> errs > results` : Redirects errors to one file (errs) and standard

output to another (results).

### 4. Piping and Utilities

- **Pipe Line (|):** Used to send the standard output of one command as the standard input to another. Example: `$ ls -lR / | more`.
- **tee Command:** Reads from standard input and writes simultaneously to standard output and a specified file. Example: `$ ls -lR / | tee fname | more`.
- **Text Processing Commands:**
  - **wc (Word Count):** Displays the number of characters, words, and lines in a file.
    - Options: `-c` (characters), `-l` (lines), `-w` (words).
  - **diff:** Compares the contents of two files, showing differences.
  - **grep:** Displays lines of input that match a specified pattern (regular expression).
    - Options: `-i` (ignore case), `-l` (list file name), `-n` (precede line with number), `-v` (inverse search/non-matching lines), `-w` (search for complete word).
  - **tr (Translate):** Translates characters from standard input to standard output.
  - **cut:** Cuts fields or columns of text. Options: `-f` (field/column), `-d` (delimiter, default TAB), `-c` (cuts by characters).
  - **sort:** Sorts text data from file or command output to standard output.

## V. Day 5: Filesystems, Links, Packages, and Archiving

### 1. Inodes and Filesystem Structure

- **Inode (Index Node):** Linux identifies all files by numerical index nodes.
- **Inode Table:** A table within each filesystem that maps used inodes to specific files.
- **Inode Stored Information (Metadata):** The inode table stores file metadata, including: file type, permissions, number of links, owner IDs (user and group), timestamps (last changed/accessed), and the physical location of the file content

on the media.

- **Viewing Inode Number:** Use `ls -i fname`.

### 2. File Manipulation and Links

- **cp vs. mv (Internal Filesystem):**
  - Copy (cp): Allocates a new inode number for the copy, creating a completely independent file.
  - Move (mv): If the source and destination are on the same filesystem, mv simply creates a new directory entry referencing the original inode number and deletes the old entry.
- **Soft Link (Symbolic Link):**
  - Creation: `ln -s testfile testlink`.
  - Mechanism: Creates a new inode entry. The content of this new entry is the path to the original file.
  - Boundary: Can be used across partition boundaries.
  - Note: If the original file is deleted, the soft link becomes an "orphaned link".
- **Hard Link:**
  - Creation: `ln testfile testlink`.
  - Mechanism: Creates a new directory entry that points directly to the exact same inode as the original file. The file content exists only once.
  - Link Count: The link count in the inode table is incremented.
  - Boundary: Cannot reach across partition boundaries; must exist within a single partition.
  - Note: The file content remains accessible and stored until the link count reaches zero (i.e., all hard links are removed).

### 3. Disk Space Management

- **df (Disk Free):** Displays the number of free disk blocks and files.
  - Syntax: `df [-h] [block_device| directory|file]`.
  - Option: `-h` provides human-readable sizes.
- **du (Disk Usage):** Displays the total sum of space allocated to all files rooted in a specified directory.

- Syntax: `du [-sh] [dir…] .`
- Option: `-s` provides a summary.

### 4. Package Management Systems

- **RPM (Red Hat Package Manager):** Installs packages from *.rpm files. RPMs are digitally signed with a GPG key by the vendor for trustworthiness.
  - `rpm -i somefile.rpm` : Install.
  - `rpm -e somefile` : Remove.
  - `rpm -U somefile` : Upgrade (removes the old version).
  - `rpm -F somefile` : Update (keeps the old version).
  - `rpm -qa` : Query all installed packages.
- **YUM (Yellowdog Updater, Modified):** A front-end to rpm.
  - Advantage: Resolves most dependencies automatically and seeks the most recent updates from online repositories. Repository locations are configured in /etc/yum.repos.d/*.repo.
  - `yum search somefile` : Look for a package.
  - `yum install somefile` : Install package and dependencies.
  - `yum remove somefile` : Uninstall package.
  - `yum upgrade somefile` : Upgrade package, removing prior versions.
  - `yum update somefile` : Update package, keeping prior versions.
  - `yum clean all` : Cleans yum download directories.

### 5. Finding Files

- **locate command:** Searches through a pre-built database of the filesystem contents.
  - Database: Built using the `updatedb` command.
  - Speed/Accuracy: Faster than find, but the information is only as current as the last database update.
- **find command:** Searches the live filesystem.
  - Speed/Accuracy: Slower than locate, causes more load, but is more powerful and searches current data. Limited by user permissions.

- Key Expressions: `-name filename` (finds by name, supports wildcards in quotes), `-size [+|-]n` (finds files larger, smaller, or exactly n 512-byte blocks), `-mtime [+|-]n` (modified time), `-user loginID`, `-type` (e.g., f for file, d for directory), `-perm` (permissions).

### 6. Archiving and Compression

Archiving is used to create copies of files/directories (archives) to safeguard them.

- **tar (Archiving):** Archives files to/extracts files from a single tar file.
  - Key Functions/Options:
    - `c` : Create a new tar file.
    - `t` : List table of content.
    - `x` : Extracts files.
    - `f` : Specifies the archive file (required).
    - `v` : Verbose mode (shows files processed).
  - Example: `tar cvf file.tar file1 file2` (create). `tar xvf file.tar` (extract).
- **Compression Tools:** Reduce file size.

| Command | Function | Output Extension | Viewing Command | Restoring Command |
|---|---|---|---|---|
| compress | Reduces text file size by 50-60%. | .Z. | zcat. | uncompress. |
| gzip | Reduces file size. | .gz. | gzcat. | gunzip. |
| bzip2 | Reduces file size. | .bz2. | bzcat. | bunzip2. |
| zip | Compresses multiple files into a single archive. | .zip. | unzip -l (list). | unzip (restore). |

## Quick Revision Section

| Concept | Term/Formula | Explanation/Definition |
|---|---|---|
| Command Syntax | command [options] [arguments] | Structure of most Linux commands. |
| FOSS License | GNU GPL | The most common public license for Free/Open Source Software. |
| Linux Core | Kernel | The core of the OS; stays resident in RAM; manages hardware/drivers. |
| Shell Interface | bash | Bourne Again Shell; the most common interface for communicating with the kernel. |
| Process Priority | Niceness Range: -20 to +19 | Smaller numbers are higher priority. Root can set priority up to -20. |
| Process Management | SIGTERM (15) / SIGKILL (9) | Standard/Default signal to terminate a process (15). Forceful, uncatchable termination signal (9). |
| Documentation | man -k keyword | Searches manual page descriptions for keywords. |
| Redirection | Command 2> errs | Redirects Standard Error (2) to the file errs. |
| Piping | Command1 \| Command2 | Pipes output from Command1 as input to Command2. |
| File Identification | Inode | A number used by Linux to track metadata about a file. |
| Soft Link | ln -s target link | A separate file entry containing the path to the original file; can cross filesystems; deleting target results in orphaned link. |
| Hard Link | ln target link | A new directory entry referencing the original file's inode; cannot cross filesystems; link count tracks removal. |

| | | |
|---|---|---|
| Package Query | rpm -qa or yum list installed | Lists all packages currently installed on the system. |
| Filesystem Search | find / -name "file.txt" | Searches the live filesystem (slower, more powerful). |
| Database Search | locate file.txt | Searches a pre-built database (faster, potentially outdated). |
| Permissions | Octal 755 (rwxr-xr-x) | Owner has Read, Write, Execute (7). Group and Others have Read, Execute (5). |
| Default Permissions | umask | Command that sets default permissions for new files and directories. |
| Vi Quit/Save | :wq or ZZ (Last Line mode) | Commands to write (save) and quit the Vi editor. |
| Vi Force Quit | :q! (Last Line mode) | Command to quit Vi without saving changes. |

End of Red Hat System Administration I (RHSA1) Study Guide Summary