Chapter 1: Improving Command-line Productivity

Concepts:

- **Bash Shell:** The command-line interpreter used in Red Hat Enterprise Linux. It provides features like command history, tab completion, and scripting capabilities.
- **Shell Scripts:** Text files containing sequences of commands that can be executed together to automate tasks. They improve efficiency and accuracy for routine tasks.
- **Command Chaining:** Connecting multiple commands using pipes (|) to pass the output of one command as the input of another.
- **Efficiency and Accuracy:** Utilizing Bash features and scripting enhances the speed and reliability of system administration tasks.
- **Automation:** Shell scripts allow for the automation of repetitive and complex tasks, reducing manual effort.
- **Pattern Matching (File Globbing):** A command-line parsing technique for specifying multiple file names easily using metacharacters like *. Primarily used for file-name patterns on the command line.
- **Regular Expressions:** Powerful patterns used to match text in files and command output. They have their own set of metacharacters and rules for pattern interpretation, which can differ from pattern matching.

Commands:

- **Creating a Bash script:** Open a new empty file in a text editor (like vim or emacs). The first line should specify the command interpreter using the **shebang notation #!/bin/bash**.
- **Executing a Bash script:** Save the script and make it executable using the chmod +x <script_name> command. Run it by specifying its path, e.g., ./<script_name>.
- **Redirecting output:**
    - >: Redirects the output of a command to a file, overwriting the file if it exists.
    - >>: Appends the output of a command to a file.
- **echo:** Displays a line of text. Used in scripts to provide output or create files with specific content.
- **Loops (for):** A control flow structure to iterate over a list of items and execute commands for each item.
- for VARIABLE in item1 item2 ... itemN
- do
- command(s)
- done
- **Conditionals (if/then/fi):** Allow decision-making in scripts, executing certain portions only when specific conditions are met.

- if <CONDITION>; then
- <STATEMENT>
- fi

elif and else can be added for more complex conditions.

- **test (or [and]):** Used to evaluate conditions in conditional statements and loops.
    - Numeric comparisons: -eq, -ne, -gt, -ge, -lt, -le.
    - String tests: -z (zero length), -n (non-zero length), =, !=.
    - File tests: -e (exists), -f (regular file), -d (directory).
    - **Important:** Spaces inside the test brackets [ ] are mandatory. The [is an alias for the test command.
- **grep:** Filters lines in a file or command output that match a given regular expression.
    - -i: Ignore case sensitivity.
    - -v: Only display lines that do *not* contain matches.
    - -r: Recursively search through directories.
    - -A NUMBER: Display NUMBER of lines after the match.
    - -B NUMBER: Display NUMBER of lines before the match.
    - -e REGEXP: Use REGEXP as the regular expression; can be used multiple times with a logical OR.
- **Regular Expression Metacharacters (examples):**
    - .: Matches any single character (except newline).
    - *: Matches the preceding item zero or more times.
    - ?: Matches the preceding item zero or one time.
    - +: Matches the preceding item one or more times.
    - {n}: Matches the preceding item exactly n times.
    - {n,}: Matches the preceding item n or more times.
    - {,m}: Matches the preceding item at most m times.
    - {n,m}: Matches the preceding item at least n times, but not more than m times.
    - []: Matches any single character within the brackets. Use [^] to match any character *not* within the brackets.
    - ^: Matches the beginning of a line.
    - $: Matches the end of a line.
    - \<: Matches the beginning of a word.

    - \>: Matches the end of a word.
- **Character Classes (within []):** Provide a way to match categories of characters. Examples include [:alnum:] (alphanumeric), [:alpha:] (alphabetic), [:digit:] (digits), [:space:] (space characters).

Important Points:

- The **#!/bin/bash (shebang) line** is crucial for indicating that the script should be executed with the Bash interpreter.
- You need to use chmod +x <script_name> to **make a script executable**.
- The **exit code** of a command or script can be checked using $?. A value of 0 typically indicates success.
- **Spaces are mandatory** inside the square brackets [ ] used with the test command.
- **Pattern matching (globbing) and regular expressions are different** systems with potentially overlapping but distinct metacharacters and rules. Regular expressions are more powerful for complex text searching.

Chapter 2: Scheduling Future Tasks

Concepts:

- **Deferred Jobs/Tasks:** Jobs scheduled to run once at some point in the future.
- **Recurring User Jobs:** Tasks that a user schedules to execute on a repeating schedule. Configured using the user's **crontab file**.
- **Recurring System Jobs:** Administrative tasks with system-wide impact that are executed on a repeating schedule. Configured using the **system crontab file (/etc/crontab)** and **system cron directories (/etc/cron.daily, /etc/cron.weekly, /etc/cron.monthly)**.
- **Systemd Timers:** Systemd unit files (.timer) that can be used to trigger other systemd units (like service units) at specified times or intervals. They can execute both deferred and recurring jobs.
- **Temporary File Management:** Mechanisms to automatically clean up temporary files and directories based on defined rules.

Commands:

- **at:** Schedules a command to run once at a specified time.
    - at now +<minutes|hours|days>
    - at HH:MM [YYYY-MM-DD]
    - Input commands after the at command and press **Ctrl+D** to finish.
    - Use input redirection to run a script: at now +5min < myscript.
- **atq:** Displays the list of pending at jobs.
- **atrm <job_id>:** Deletes a specific at job.

- **crontab -e:** Opens the user's crontab file in an editor to create or modify scheduled jobs.
- **crontab -l:** Lists the current user's crontab entries.
- **crontab -r:** Deletes the current user's crontab.
- **Cron Job Syntax (in crontab files):**
- minute hour day_of_month month day_of_week command_to_run

Each field can contain a specific value, a range (e.g., 1-5), a list (e.g., 1,3,5), or an asterisk (*) to match all values.

- **System Crontab File (/etc/crontab):** Similar to user crontabs but includes an additional **user field** specifying which user to run the command as.
- **System Cron Directories (/etc/cron.daily, /etc/cron.weekly, /etc/cron.monthly):** Scripts placed in these directories are executed daily, weekly, or monthly, respectively, by the anacron utility or systemd timers.
- **systemctl enable <timer_unit>.timer:** Enables a systemd timer unit to start on boot.
- **systemctl disable <timer_unit>.timer:** Disables a systemd timer unit.
- **systemctl start <timer_unit>.timer:** Starts a systemd timer unit immediately.
- **systemctl stop <timer_unit>.timer:** Stops a systemd timer unit.
- **systemctl status <timer_unit>.timer:** Shows the status and details of a systemd timer unit.
- **systemd-tmpfiles --create:** Creates temporary files and directories based on the configuration files in /etc/tmpfiles.d/ and /run/tmpfiles.d/.
- **Temporary File Configuration Files (/etc/tmpfiles.d/*.conf):** Define rules for creating, deleting, and managing temporary files and directories. Each line specifies an action (d for directory, f for file, D for empty directory, etc.), the path, permissions, ownership, and age/expiration time.

Important Points:

- at is for **one-time execution** of commands in the future.
- cron is for **recurring tasks** scheduled by users and the system. User-specific tasks are managed with crontab -e, while system-wide tasks are often defined in /etc/crontab or the cron directories.
- **Systemd timers are the modern way to schedule tasks** in Red Hat Enterprise Linux and offer more flexibility and integration with the systemd ecosystem.
- Temporary files are crucial for system operation, and **systemd-tmpfiles** ensures their proper management, including creation and cleanup based on configuration files.

Chapter 3: Tuning System Performance

Concepts:

- **System Performance Optimization:** Adjusting various system parameters and settings to improve the efficiency and responsiveness of the system.

- **Tuning Profiles:** Pre-defined sets of configuration changes managed by the **tuned daemon** to optimize performance for specific workloads (e.g., throughput, latency, power saving).
- **tuned Daemon:** A service that automatically modifies device settings based on the active tuning profile.
- **Process Scheduling:** The kernel's mechanism for deciding which processes get access to the CPU and for how long.
- **Priority:** A relative value assigned to a process to influence its CPU access.
- **Nice Value:** A user-space representation of a process's scheduling priority. Ranges from **-20 (highest priority)** to **19 (lowest priority)**. Higher nice values mean lower priority.
- **Multitasking:** The ability of an operating system to execute multiple processes concurrently.

Commands:

- **systemctl enable tuned:** Starts the tuned service and configures it to start on boot.
- **systemctl start tuned:** Starts the tuned service immediately.
- **systemctl status tuned:** Checks the status of the tuned service.
- **tuned-adm active:** Displays the currently active tuned profile.
- **tuned-adm list:** Lists the available tuned profiles.
- **tuned-adm profile <profile_name>:** Switches the active tuned profile to the specified one. This applies the performance tuning settings of that profile.
- **tuned-adm off:** Deactivates the tuned service, reverting system settings to their defaults (before tuned was active).
- **nice [OPTIONS] [COMMAND [ARG]...]:** Runs a command with a specified nice value, setting its initial priority.
    - nice -n <value> <command> (e.g., nice -n 10 sha1sum /dev/zero &)
    - If no value is specified, the nice value is incremented by 10.
- **renice <priority> <PID>...:** Modifies the nice value (priority) of one or more running processes identified by their Process IDs (PIDs).
    - renice -10 1234 (sets the nice value of process 1234 to -10, increasing its priority if the user has sufficient privileges).

Important Points:

- The **tuned service** allows for easy optimization of system performance by selecting appropriate profiles for different use cases.
- Activating a tuning profile automatically adjusts various system settings.
- You can switch to another profile or deactivate tuned to **revert changes** made by a profile.
- The **nice command** sets the initial scheduling priority of a new process.

- The **renice command** allows you to change the scheduling priority of a process that is already running.
- Lower nice values give a process **higher priority** for CPU access, while higher nice values give it lower priority. Only the root user can decrease the nice value (increase priority) of a process.

Chapter 4: Controlling Access to Files with ACLs

Concepts:

- **Access Control Lists (ACLs):** A more flexible mechanism for controlling file and directory permissions beyond the traditional owner, group, and others. ACLs allow you to define permissions for specific users or groups.
- **Standard Linux File Permissions:** The traditional read (r), write (w), and execute (x) permissions for the owner, group, and others.
- **ACL Entries:** Individual permission settings within an ACL. Common types include:
    - **User entries:** Permissions for a specific named user (user:<username>:<permissions>).
    - **Group entries:** Permissions for a specific named group (group:<groupname>:<permissions>).
    - **Mask entry:** Defines the maximum permissions allowed for named user and group entries (excluding the file owner and group owner).
    - **Owner entry:** Permissions for the file owner (user::<permissions>).
    - **Group owner entry:** Permissions for the file's group owner (group::<permissions>).
    - **Other entry:** Permissions for all other users (other::<permissions>).
    - **Default ACLs:** ACL entries that are automatically applied to new files and subdirectories created within a directory. They start with default: (e.g., default:user:mary:rx).
- **ACL Permission Precedence:** When an ACL is present, it takes precedence over the standard group permissions for named users and groups.
- **File-system Mount Options:** Certain mount options are required for ACLs to be supported on a file system. Most modern Linux file systems (like XFS and ext4) support ACLs by default, but it's essential to ensure the acl mount option is enabled.

Commands:

- **getfacl <file/directory>:** Displays the ACL entries for the specified file or directory. It shows the owner, group, and the ACL entries, including the mask.
- **setfacl [OPTIONS] <file/directory>:** Sets, modifies, or removes ACL entries on files and directories.
    - -m <acl_entry>: Modifies or adds the specified ACL entry (e.g., setfacl -m u:john:rw file.txt). Multiple entries can be specified, separated by commas.

- -x <acl_entry>: Removes the specified ACL entry (e.g., setfacl -x g:developers directory).
- -b: Removes all ACL entries from a file or directory, reverting to standard permissions.
- -d -m <default_acl_entry> <directory>: Sets a default ACL entry for a directory (e.g., setfacl -d -m g:staff:rwx shared_folder).
- --set <acl_entries> <file/directory>: Replaces the entire ACL with the specified entries.
- --set-file=<file> <file/directory>: Sets the ACL based on entries in a file.
- -R: Applies changes recursively to all files and subdirectories within a directory.

Important Points:

- ACLs provide **fine-grained access control** beyond the basic owner, group, and others permissions.
- Use getfacl to **view and interpret existing ACLs**.
- Use setfacl to **set, modify, and remove both standard and default ACLs**.
- **Default ACLs** are crucial for automatically managing permissions for new files and subdirectories within a shared directory.
- The **ACL mask** limits the effective permissions of named users and groups. When setting ACLs, be mindful of the mask; you might need to adjust it explicitly to grant the desired permissions to named users and groups.
- Red Hat Enterprise Linux uses ACLs by default in some cases, such as on removable devices, to manage permissions dynamically.

Chapter 5: Managing SELinux Security

Concepts:

- **Security-Enhanced Linux (SELinux):** A Linux kernel security module that provides a mandatory access control (MAC) system. It enhances system security by enforcing policies that define how processes can interact with files and other system objects.
- **Mandatory Access Control (MAC):** A security model where access is determined by system-wide policies, in addition to the traditional discretionary access control (DAC) based on user and group ownership and permissions.
- **SELinux Modes:**
  - **Enforcing:** SELinux actively enforces the security policy, denying actions that violate the policy and logging these denials.
  - **Permissive:** SELinux does not deny actions but logs all policy violations. Useful for troubleshooting and transitioning to enforcing mode.
  - **Disabled:** SELinux is completely turned off.

- **SELinux Contexts (Labels):** Every process, file, directory, and other system object is labeled with an SELinux context. A typical context includes user, role, type, and sensitivity (e.g., user_u:object_r:httpd_sys_content_t:s0). The **type** is the most important part for policy enforcement, defining the object's function.
- **SELinux Policy:** A set of rules that define how processes with specific contexts can interact with objects with other contexts.
- **SELinux Booleans:** Policy rules that can be toggled on or off at runtime, allowing for selective adjustments to the SELinux policy without requiring relabeling or policy recompilation.
- **SELinux AVC (Access Vector Cache) Denials:** Messages logged when SELinux prevents an action because it violates the loaded security policy. These messages are crucial for troubleshooting SELinux issues.

Commands:

- **getenforce:** Displays the current SELinux mode (Enforcing, Permissive, or Disabled).
- **setenforce <0|1>:** Changes the current SELinux mode. 0 for Permissive, 1 for Enforcing. This change is temporary and will not persist across reboots.
- **Editing /etc/selinux/config:** This file sets the default SELinux mode that the system will boot into. Modify the SELINUX variable (e.g., SELINUX=enforcing).
- **ls -Z <file/directory>:** Displays the SELinux context of the specified file or directory.
- **ps -Z <PID>:** Displays the SELinux context of the specified process.
- **semanage fcontext -l [file_spec]:** Lists the SELinux file context definitions from the policy. You can filter by a file specification.
- **semanage fcontext -a -t <type> <file_spec>:** Adds a new file context mapping rule to the SELinux policy, associating files matching <file_spec> with the type <type>. This only modifies the policy, not the existing file contexts.
- **restorecon -v <file/directory>...:** Applies the file contexts defined in the SELinux policy to the specified files and directories. Use -R for recursive application. -v for verbose output. This command relabels the files.
- **semanage boolean -l [| grep <keyword>]:** Lists the available SELinux booleans and their current states (on/off). You can use grep to filter the list.
- **getsebool <boolean_name>:** Displays the current state (true/false) of a specific SELinux boolean.
- **setsebool [-P] <boolean_name> <on|off>:** Changes the state of an SELinux boolean. Use -P to make the change persistent across reboots (by writing to the policy).
- **ausearch -m avc [-ts recent]:** Searches the audit log for SELinux Access Vector Cache (AVC) denial messages. -ts recent can filter for recent events.
- **sealert -a /var/log/audit/audit.log:** Analyzes the audit log and provides explanations and potential solutions for SELinux denials.

Important Points:

- SELinux operates based on **labels (contexts)** and a **security policy** that defines allowed interactions.
- Understanding the **SELinux mode** is crucial for troubleshooting. In enforcing mode, violations are prevented, while in permissive mode, they are only logged.
- The **semanage fcontext and restorecon commands** are used to manage and apply SELinux file contexts. You modify the policy with semanage and apply it to the file system with restorecon.
- **SELinux booleans** provide a way to customize the policy at runtime without full relabeling.
- When troubleshooting access issues, check for **SELinux AVC denials** in the audit log using ausearch or sealert.

Chapter 6: Managing Basic Storage

Concepts:

- **Disk Partitioning:** Dividing a physical storage device into multiple logical storage units called partitions. Each partition can be formatted with a file system or used for swap space.
- **Partitions (Primary, Extended, Logical):** On MBR disks, there can be up to four primary partitions, or three primary and one extended partition. An extended partition can contain multiple logical partitions. GPT disks do not have this distinction and can support a large number of partitions.
- **Partition Tables (MBR, GPT):** Structures on a disk that contain information about the partitions. **MBR (Master Boot Record)** is older and has limitations, while **GPT (GUID Partition Table)** is newer and more flexible.
- **File Systems:** Structures that organize data on a partition, allowing the operating system to store and retrieve files (e.g., XFS, ext4).
- **Formatting:** The process of creating a file system on a partition, making it usable for storing data.
- **Mounting:** Making a file system on a partition accessible within the directory tree at a specific mount point.
- **Persistent Mounts:** Configuring file systems to be automatically mounted at boot time by adding entries to the /etc/fstab file.
- **Swap Space:** An area on disk (either a dedicated partition or a file) used to extend the amount of available RAM when the physical memory is full.
- **Swap Partitions:** Dedicated partitions formatted as swap space.
- **Swap Files:** Regular files used as swap space.

Commands:

- **parted /dev/<device>:** A disk partitioning utility that can work with various partition table types, including GPT and MBR.
    - mklabel <type>: Creates a new disk label (e.g., mklabel gpt).

    - mkpart [primary|logical|extended] <filesystem_type> <start> <end>: Creates a new partition. Units can be MB, GB, %, etc.
    - print: Displays the current partition table.
    - resizepart <partition_number> <start> <end>: Resizes an existing partition.
    - set <partition_number> <flag> on|off: Sets or clears partition flags (e.g., set 1 boot on).
- **mkfs.<type> /dev/<partition>:** Creates a file system of the specified type on a partition (e.g., mkfs.xfs /dev/sdb1, mkfs.ext4 /dev/sdb2).
- **mount /dev/<device> <mountpoint>:** Mounts the file system on the specified device to the given mount point.
    - Without arguments, mount displays the currently mounted file systems.
- **umount <mountpoint> or umount /dev/<device>:** Unmounts a file system.
- **Editing /etc/fstab:** The configuration file for persistent mounts. Each line defines a mount point with the following format:
- <file_system> <mount_point> <type> <options> <dump> <pass>
    - <file_system>: The device file or UUID/LABEL of the partition.
    - <mount_point>: The directory where the file system will be mounted.
    - <type>: The file system type (e.g., xfs, ext4, nfs, swap).
    - <options>: Mount options (e.g., defaults, ro, noatime, acl). Use defaults for common options. For swap, use sw.
    - <dump>: Used by the dump utility (usually 0).
    - <pass>: Order for file system checks at boot (root is 1, others are usually 2, swap is 0).
- **mkswap /dev/<partition> or mkswap /path/to/swapfile:** Initializes a partition or file as swap space.
- **swapon /dev/<partition> or swapon /path/to/swapfile:** Enables a swap space for use. Use -a to enable all swap spaces listed in /etc/fstab.
- **swapoff /dev/<partition> or swapoff /path/to/swapfile:** Disables a swap space.
- **swapon -s or cat /proc/swaps:** Displays information about active swap spaces.

Important Points:

- **Partitioning** allows for organizing storage and using different file systems or swap. Choose between MBR and GPT based on requirements (size, number of partitions).
- **Formatting** creates a usable file system on a partition.
- **Mounting** makes the file system accessible.

- **/etc/fstab** is critical for ensuring file systems and swap are automatically activated at boot. Use UUIDs or LABELS for robustness.
- **Swap space** is used to extend memory. You can create swap partitions or swap files. Use swapon and swapoff to manage active swap space.

Chapter 7: Managing Logical Volumes

Concepts:

- **Logical Volume Management (LVM):** A flexible storage management system that allows you to abstract physical storage devices into logical volumes, providing features like resizing, snapshots (not in the initial chapters), and striping.
- **Physical Volumes (PVs):** Physical storage devices (partitions or whole disks) that are initialized for use by LVM.
- **Volume Groups (VGs):** Collections of one or more PVs. A VG creates a pool of storage from which Logical Volumes can be created.
- **Logical Volumes (LVs):** Virtual block devices created from the free space within a VG. LVs can be formatted with file systems or used as swap space.
- **Physical Extents (PEs):** Fixed-size blocks into which PVs are divided.
- **Logical Extents (LEs):** Corresponding fixed-size blocks in LVs that map to PEs. By default, one LE maps to one PE.

Commands:

- **pvcreate /dev/<device>...:** Initializes one or more devices as Physical Volumes for use by LVM.
- **pvdisplay [/dev/<device>]:** Displays information about PVs. Without an argument, it shows details of all PVs.
- **vgcreate <vg_name> /dev/<pv>...:** Creates a new Volume Group named <vg_name> using the specified PVs.
- **vgdisplay [<vg_name>]:** Displays information about VGs. Without an argument, it shows details of all VGs.
- **lvcreate -L <size>[m|g|t] -n <lv_name> <vg_name>:** Creates a new Logical Volume named <lv_name> of the specified size within the <vg_name>. Use -l <number> to specify the size in logical extents.
- **lvdisplay [/dev/<vg_name>/<lv_name>]:** Displays information about LVs. Without an argument, it shows details of all LVs.
- **lvextend -L [+]<size>[m|g|t] /dev/<vg_name>/<lv_name>:** Extends the size of an existing LV. Use + before the size to add to the current size.
- **lvreduce -L [-]<size>[m|g|t] /dev/<vg_name>/<lv_name>:** Reduces the size of an existing LV (caution: can lead to data loss if the underlying file system is larger than the new LV size).

- **resize2fs /dev/<vg_name>/<lv_name> [<new_size>]:** Resizes an ext2/3/4 file system to match the size of its underlying LV. Should be run after lvextend or lvreduce. If <new_size> is not specified, it will resize to the current LV size.
- **xfs_growfs <mountpoint>:** Resizes an XFS file system to match the size of its underlying LV. The file system must be mounted.
- **vgextend <vg_name> /dev/<pv>...:** Adds one or more PVs to an existing VG, increasing its capacity.
- **vgreduce <vg_name> /dev/<pv>...:** Removes one or more PVs from a VG, decreasing its capacity (PV must not contain any LVs).
- **pvmove /dev/<pv_old> /dev/<pv_new>:** Moves the physical extents from one PV to another (within the same VG). Useful for replacing disks.
- **lvremove /dev/<vg_name>/<lv_name>:** Removes a Logical Volume, freeing up its space in the VG. The LV should be unmounted first.
- **vgremove <vg_name>:** Removes an empty Volume Group (no LVs present).
- **pvremove /dev/<pv>:** Removes a PV from LVM management (no longer part of any VG).

Important Points:

- LVM provides **flexible storage management** by creating a layer of abstraction between physical disks and logical volumes.
- The key components are **Physical Volumes (PVs)**, **Volume Groups (VGs)**, and **Logical Volumes (LVs)**.
- You can **extend the size of LVs** and their underlying file systems non-destructively, providing the VG has enough free space.
- LVM allows you to **pool storage from multiple physical devices** into a single, manageable VG.
- Logical volumes can be **resized dynamically** while the file system is still mounted and in use (for extension with XFS). Remember to resize the file system after extending the LV.