Red Hat System Administration II (RH134)

Chapter Title: Improving Command-line Productivity

- Key Concepts:
    - Using advanced features of the Bash shell, shell scripts, and various utilities provided by Red Hat Enterprise Linux to run commands more efficiently.
    - Automating sequences of commands by writing simple shell scripts.
    - Chaining multiple commands that pass results between them.
    - Improving the efficiency and accuracy of routine task completion through script use.
    - Creating Bash shell scripts using text editors like vim or emacs, which can provide syntax highlighting to identify errors.
    - Specifying the command interpreter: The first line of a script begins with the notation #! (sh-bang or she-bang), which is a two-byte magic number indicating an interpretive script.
    - Quotation marks: Use single quotation marks to interpret all text literally, suppressing globbing, shell expansion, command substitution, and variable substitution. Metacharacters like ? also need protection from expansion.
    - Iterating over lists using for loops.
    - Evaluating exit codes from commands and scripts. A zero exit code typically indicates success.

    - Using conditional structures like the if/then construct to incorporate decision making into shell scripts, executing parts of the script only when certain conditions are met.
    - Using the test command for evaluating conditional expressions. The bash(1) man page also explains operator use and evaluation.
    - Regular expressions: Provide a pattern matching mechanism for finding specific content. They are a language of their own with syntax and rules.
    - An exact match is the simplest regular expression, matching characters in the regular expression to the type and order in the searched data.
    - Regular expressions are designed to represent any form or pattern in text strings, no matter how complex. They are internally supported by numerous text processing commands like grep, sed, awk, and programming languages like Python and Perl.
    - Common regular expression metacharacters include . (any single character), ? (preceding item optional, matched at most once), * (preceding item matched zero or more times), and + (preceding item matched one or more times).
    - Character classes ([:alnum:], [:alpha:], [:blank:], etc.) can be used within regular expressions.
    - Pattern matching (globbing or file-name expansion) is a command-line parsing technique primarily for specifying many file names; it uses similar metacharacters to regular expressions but has different interpretation rules.
- Important Commands:

    - echo: Used in scripts to print output.

- hostname -s: Example command used within variable substitution.
- cat: Used to review the content of files, such as script output or configuration files.
- vim: A text editor recommended for creating and editing scripts due to syntax highlighting.
- rm: Used to remove files.
- chmod a+x <script_file>: Makes a script file executable.
- ssh <user>@<host>: Used to log in to remote systems to run commands.
- lscpu: Command whose output can be filtered.
- grep: Apply regular expressions to text files, search files and data from piped commands.
    - -i: Ignore case distinctions.
    - -v: Invert match, only display lines that *do not* contain matches.
    - -r: Apply search recursively to files/directories.
    - -A NUMBER: Display NUMBER lines after match.
    - -B NUMBER: Display NUMBER lines before match.
    - -e: Use multiple regular expressions with logical OR.
- test: Evaluate conditional expressions in scripts. Refer to its man page for operators.
- lab console-write start/finish: Lab commands for a guided exercise on writing scripts.
- lab console-commands start/finish: Lab commands for a guided exercise on using loops.
- lab console-regex start/finish: Lab commands for a guided exercise on using regular expressions with grep.
- lab console-review start/grade/finish: Lab commands for the chapter lab.

- Applications:

    - Automating routine system administration tasks by writing simple shell scripts containing sequences of commands.
    - Using for loops to efficiently run commands over lists of items from the command line or in scripts.
    - Implementing decision making in scripts using conditional structures based on command exit codes or other conditions.
    - Filtering relevant content from files (like logs or configuration files) or command output using grep and regular expressions.
    - Combining commands and filtering to collect specific information from multiple remote hosts.

Chapter Title: Scheduling Future Tasks

- Key Concepts:

    - Deferred jobs or tasks: Commands scheduled to run once at some point in the future.
    - Recurring user jobs: Commands scheduled to run on a repeating schedule using a user's crontab file.
    - Recurring system jobs: Administrative tasks scheduled to run on a repeating schedule using the system crontab file and directories.
    - Crontab file format: Each job is typically on a single line, with fields for minute, hour, day of month, month, day of week, user-name (for system crontab), and command.

    - Special entries in crontab include empty lines, comments (starting with #), and environment variables (e.g., SHELL, MAILTO).
    - The command is executed when *either* the Day of month or Day of week field (if both are not *) is satisfied.
    - System crontab files: /etc/crontab and files within /etc/cron.d/, /etc/cron.hourly/, /etc/cron.daily/, /etc/cron.weekly/, and /etc/anacrontab.
    - Systemd timer units: Can execute both deferred and recurring jobs.
    - Managing Temporary Files: Modern systems use /tmp and volatile directories under /run for temporary data.
    - systemd-tmpfiles: A utility for managing temporary files.
    - tmpfiles.d configuration files: Define how systemd-tmpfiles manages temporary files, including removal policies.
    - The systemd-tmpfiles-clean.timer unit regularly triggers the cleanup of temporary files.
- Important Commands:

    - at TIMESPEC: Schedule a new job to run once. Reads commands from standard input; input can be ended with Ctrl+D or redirected from a file (< myscript).
    - atq: List scheduled jobs. (Implied by Quiz question 1, source does not explicitly list it with usage).
    - crontab -e: Edit the user's crontab file. Invokes Vim by default unless EDITOR is set.
    - crontab -l: Display all the user jobs currently scheduled.
    - crontab -r: Remove the user's crontab file.
    - crontab -u <user>: View or modify another user's crontab.

    - systemd-tmpfiles --clean: Manually trigger cleanup defined in tmpfiles.d.
    - systemd-tmpfiles --remove: Manually trigger removal defined in tmpfiles.d.
    - systemctl list-timers: List active timers.
    - systemctl status systemd-tmpfiles-clean.timer: Check the status of the temporary file cleanup timer.
    - lab scheduling-system start/finish: Lab commands for a guided exercise on scheduling system jobs.
    - lab scheduling-tempfiles start/finish: Lab commands for a guided exercise on managing temporary files.
- Applications:

    - Scheduling commands to run once in the future for delayed execution.
    - Setting up commands to run automatically on a regular basis for a specific user using crontab.
    - Configuring system-wide administrative tasks to run on a repeating schedule using system crontab files.
    - Managing the cleanup of temporary files and directories on the system.
    - Configuring the frequency and policies for removing temporary files using systemd-tmpfiles and tmpfiles.d.

Chapter Title: Tuning System Performance

- Key Concepts:

    - Improving system performance by setting tuning parameters and adjusting scheduling priority of processes.
    - The tuned daemon automatically modifies device settings to meet specific system needs based on a pre-defined selected tuning profile.
    - Tuning profiles are managed by the tuned daemon.
    - To revert changes made by a selected profile, switch to another profile or deactivate the tuned service.
    - The system assigns a relative priority to a process, called the nice value, to determine its CPU access. A higher nice value means lower priority (less CPU time).
    - Influencing Process Scheduling: Prioritizing or de-prioritizing specific processes.
- Important Commands:

    - tuned-adm list: List available tuning profiles.
    - tuned-adm profile <profile_name>: Activate a specific tuning profile.
    - tuned-adm active: Show the currently active tuning profile.
    - tuned-adm profile_info: List summary information of the current active profile.
    - nice <command>: Assign a priority (nice value) to a process when it starts.
    - renice <priority> <PID>: Adjust the nice level of an existing process.
    - yum list tuned: Verify if the tuned package is installed.
    - systemctl status tuned: Check the status of the tuned service.
    - systemctl is-enabled tuned: Check if the tuned service is enabled to start at boot.

    - systemctl start tuned: Start the tuned service.
    - systemctl enable tuned: Enable the tuned service to start at boot.
    - top: Command often used to identify processes consuming high CPU usage.
    - sudo: Used to run commands with root privileges.
    - ssh: Used to connect to remote servers.
    - lab tuning-profiles start/finish: Lab commands for a guided exercise on tuning profiles.
    - lab tuning-procscheduling start/finish: Lab commands for a guided exercise on process scheduling.
    - lab tuning-review start/grade/finish: Lab commands for the chapter lab.
- Applications:

    - Optimizing system performance for different workloads (e.g., desktop, server, virtual host) by applying pre-defined tuning profiles.
    - Adjusting the scheduling priority of processes to control how much CPU time they receive, allowing more resources for other processes.
    - Diagnosing and addressing performance bottlenecks related to CPU scheduling.

Chapter Title: Controlling Access to Files with ACLs

- Key Concepts:

    - Access Control Lists (ACLs): Provide fine-grained access control to files and directories, supplementing standard Linux file permissions.

    - ACLs are useful when standard permissions (owner, group, others) are insufficient for complex user and group access requirements.
    - Viewing and interpreting ACLs requires commands like ls and getfacl.
    - The ACL mask affects the maximum permissions granted to named users, named groups, and the owning group. (Specific details on interpreting the mask are not in the provided source, but the concept is mentioned).
    - ACL permission precedence determines the order in which permissions are evaluated. (Specific details on precedence are not in the provided source, but the concept is mentioned).
    - Default ACLs: Can be defined on a directory to automatically set permissions on newly created files and directories within it.
    - Red Hat Enterprise Linux uses systemd and udev to apply predefined ACLs on devices, folders, and files by default.
- Important Commands:

    - getfacl <file_or_directory>: Display the ACLs on a file or directory. Also shows the owning user, owning group, and standard permissions.
    - ls -l: Shows standard file permissions. (Implied use alongside getfacl but not detailed how it indicates ACLs are set in the source).
    - setfacl: Set, modify, and remove default and standard ACLs on files and directories. (Specific syntax examples are not provided in the source).
    - Man pages: acl(5), getfacl(1), setfacl(1), ls(1).
    - lab acl-review start/grade/finish: Lab commands for the chapter lab.
- Applications:

- Implementing access control policies that are more granular than standard Linux permissions, allowing specific users or groups access to files/directories that the owning group or others would not normally have.
- Setting default permissions for new files and directories within a specific location.
- Identifying and interpreting the access permissions granted by existing ACLs.
- Troubleshooting permission issues related to ACLs.

Chapter Title: Managing SELinux Security

- Key Concepts:
  - SELinux (Security-Enhanced Linux): Protects and manages the security of a server. It enforces mandatory access control.
  - SELinux protects resources by labeling all processes and files with a security context.
  - SELinux enforcement mode: Can be changed to enforcing, permissive, or disabled.
    - enforcing: SELinux policy is enforced, denials are logged.
    - permissive: SELinux policy is not enforced, denials are logged (useful for troubleshooting).
    - disabled: SELinux is turned off.
  - The SELinux mode can be changed temporarily at runtime or persistently via a configuration file.

    - SELinux file contexts: Control how processes interact with files. Each file and process has a label.
    - The SELinux policy rules determine the default context for files and directories.
    - semanage fcontext command manages the policy rules that define default contexts. Extended regular expressions like (/.*)? are commonly used in fcontext rules to match directories and their contents recursively.
    - restorecon command applies the context defined by the SELinux policy to files and directories.
    - SELinux Booleans: Are switches that change the behavior of the SELinux policy at runtime. They can be activated or deactivated to allow varying access needs without rewriting the policy.
    - Investigating and Resolving SELinux Issues: Includes using log analysis tools, identifying issues in system logs, and adjusting the SELinux configuration.
    - AVC denials: Messages logged by SELinux when an action is blocked by the policy.
    - sealert is a tool that displays useful information to help with SELinux troubleshooting, often suggesting potential solutions.
    - SELinux Port Labeling: Network traffic is tightly enforced by the SELinux policy. Network ports are labeled. When a process tries to listen on a port, SELinux checks if the process's label is allowed to bind that port's label.
    - semanage port command is used to add, delete, and modify port labels.
- Important Commands:

- getenforce: Display the current SELinux enforcement mode.
- setenforce 1|0|Enforcing|Permissive: Change the current SELinux mode at runtime (1 or Enforcing sets to enforcing, 0 or Permissive sets to permissive).
- grep '^SELINUX' /etc/selinux/config: Check the persistent SELinux mode setting.
- semanage fcontext: Manage SELinux policy rules for default file contexts.
  - -a, --add: Add a record.
  - -d, --delete: Delete a record.
  - -l, --list: List records.
- restorecon <file_or_directory>: Apply the context defined by policy to a file or directory.
- setsebool <boolean> on|off: Activate/deactivate SELinux policy rules using booleans temporarily.
- setsebool -P <boolean> on|off: Activate/deactivate SELinux policy rules using booleans persistently.
- semanage boolean -l: Manage or list the persistent value of SELinux booleans.
- sealert -a /var/log/audit/audit.log: Use sealert to analyze audit logs for SELinux denials.
- semanage port -a -t <port_label> -p tcp|udp <PORTNUMBER>: Add a port to an existing port label.
- semanage port -l: List default port labels.
- semanage port -d -t <port_label> -p tcp|udp <PORTNUMBER>: Delete a port label record.
- semanage -C: View local changes to the default policy.

    - lab selinux-opsmode start/finish: Lab commands for a guided exercise on changing enforcement mode.
    - lab selinux-filecontexts start/finish: Lab commands for a guided exercise on file contexts.
    - lab selinux-booleans start/finish: Lab commands for a guided exercise on booleans.
    - lab selinux-review start/finish: Lab commands for the SELinux troubleshooting lab.
- Applications:

    - Configuring the overall security posture of the system by setting the SELinux enforcement mode.
    - Ensuring that applications and services can access the files and directories they need by managing SELinux file contexts.
    - Temporarily or persistently changing specific aspects of the SELinux policy to accommodate varying access needs using booleans.
    - Diagnosing and resolving issues caused by SELinux blocking legitimate actions, using log analysis and tools like sealert.
    - Controlling network service access by verifying and adjusting SELinux port labels.

Chapter Title: Managing Basic Storage

- Key Concepts:

    - Creating and managing storage devices, partitions, file systems, and swap spaces from the command line.
    - Disk Partitioning: Dividing a hard drive into multiple logical storage units (partitions). Different partitions can serve different functions.
    - Partition Table: Indicates the partitioning scheme used on a disk, such as MBR (msdos) or GPT.
    - Writing a disk label to a new drive prepares it for a specific partitioning scheme.
    - Creating partitions using partitioning tools.
    - File Systems: Formatting partitions to store data. XFS is a commonly used file system type.
    - Mounting: Making a file system accessible at a specific point in the directory tree.
    - Persistent Mounts: Configuring file systems to be mounted automatically at boot, typically by adding entries to the /etc/fstab file.
    - Swap Space: An area of a disk used by the Linux kernel memory management to supplement physical memory.
    - Creating and managing swap spaces involves formatting a partition or file as swap and activating it.
    - Swap spaces can also be configured for persistent activation at boot.
- Important Commands:

    - Partitioning Tools:
        - parted: Add, modify, and remove partitions on disks using either MBR (msdos) or GPT schemes. Can be used in interactive mode or

with subcommands on the command line (parted <device> <command>).

- mklabel <scheme>: Write a disk label (e.g., msdos, gpt).
- print: List partitions.
- mkpart <type> <fs_type> <start> <end>: Create a new partition (type can be primary, logical, extended; fs_type can be xfs, ext4, etc.).
- name <partition_number> <name>: Assign a name to a GPT partition.
- set <partition_number> <flag> on|off: Set partition flags (e.g., boot, lvm).

- mkfs.<fs_type> <device>: Format a partition with a file system (e.g., mkfs.xfs /dev/vdb1).
- mount <device> <mount_point>: Mount a file system temporarily.
- vim /etc/fstab: Edit the file system table for persistent mounts.
- mount -a: Mount all file systems listed in /etc/fstab.
- mkswap <device_or_file>: Initialize a partition or file for use as swap space.
- swapon <device_or_file>: Activate a swap space.
- swapoff <device_or_file>: Deactivate a swap space.
- swapon --show: Display active swap spaces.
- lsblk: List block devices and their partitions.
- blkid: Locate and print block device attributes.
- cat /proc/partitions: List partitions.
- sudo -i: Switch to the root user.
- ssh: Log in to remote servers.

    - lab storage-partitions start/finish: Lab commands for guided exercise on partitions/filesystems.
    - lab storage-swap start/finish: Lab commands for guided exercise on swap.
    - lab storage-review start/grade/finish: Lab commands for the chapter lab.
- Applications:

    - Configuring new storage devices for use by creating partitions and file systems.
    - Making file systems accessible at boot time for persistent data storage.
    - Adding or modifying swap space to improve system performance when physical memory is insufficient.
    - Managing disk space effectively by dividing disks into logical units for different purposes.

Chapter Title: Managing Logical Volumes

- Key Concepts:

    - Logical Volume Management (LVM): Allows for flexible storage management by allocating space across multiple storage devices.
    - LVM makes managing disk space easier, especially when a file system needs more space.
    - LVM components:
        - Physical Volumes (PVs): Underlying storage devices (partitions or whole disks) initialized for use by LVM.

        - Volume Groups (VGs): Aggregations of one or more physical volumes, forming a pool of storage space.
        - Logical Volumes (LVs): Created from the storage pool in a volume group, act like partitions, and can be formatted with file systems or swap spaces.
    - LVM provides a set of command-line tools suitable for automation.
    - Storage can be added to or removed from volume groups.
    - The size of a logical volume formatted with a file system can be extended nondestructively.
- Important Commands:

    - LVM Tools:
        - pvcreate <device>: Initialize a device or partition as a physical volume.
        - vgcreate <vg_name> <pv1> [<pv2>...]: Create a volume group from one or more physical volumes.
        - lvcreate -L <size> -n <lv_name> <vg_name>: Create a logical volume from a volume group, specifying its size and name.
        - lvs: Display information about logical volumes.
        - vgs: Display information about volume groups.
        - pvs: Display information about physical volumes.
        - lvdisplay, vgdisplay, pvdisplay: Display detailed information about LVM components.
        - lvextend -L +<size> <lv_path>: Extend a logical volume by a specified amount.

        - xfs_growfs <mount_point>: Grow the XFS file system on a logical volume after extending the LV.
    - mkfs.xfs <lv_path>: Format a logical volume with the XFS file system.
    - vim /etc/fstab: Configure persistent mounts for logical volumes.
    - mount <lv_path> <mount_point>: Mount a logical volume.
    - parted, lsblk, blkid, cat /proc/partitions: Used to identify underlying devices for PVs.
    - ssh, sudo -i: Used to connect to remote servers and gain root privileges.
    - lab lvm-creating start/finish: Lab commands for guided exercise on creating LVM.
    - lab lvm-extending start/finish: Lab commands for guided exercise on extending LVM.
    - lab lvm-review start/grade/finish: Lab commands for the chapter lab.
- Applications:

    - Creating flexible storage pools from multiple disks or partitions.
    - Easily expanding file systems or swap spaces hosted on logical volumes as needed without repartitioning.
    - Managing storage with a higher level of abstraction than traditional partitions.

Chapter Title: Implementing Advanced Storage Features

- Key Concepts:

    - Managing storage using the Stratis local storage management system and VDO (Virtual Data Optimizer) volumes.
    - Stratis: Implements flexible file systems that grow dynamically with data. It supports thin provisioning, snapshotting, and monitoring. Stratis manages storage layers.
    - Thin provisioning: Stratis volumes initially consume minimal space and grow as data is added.
    - Snapshotting: Stratis allows creating snapshots of thin-provisioned file systems.
    - VDO (Virtual Data Optimizer): Aims to reduce the cost of data storage.
    - VDO optimizes disk space efficiency by applying zero-block elimination, data deduplication, and data compression.
- Important Commands:

    - Stratis tools (Specific commands like stratis pool create, stratis filesystem create, stratis filesystem snapshot are implied by the concepts but not detailed in the provided excerpts).
    - VDO tools:
        - vdo create --name=<name> --device=<device> --vdoLogicalSize=<size>: Create a VDO volume.
        - vdo list: Verify the availability of VDO volumes.
        - vdo status <name>: Check status and features (like compression, deduplication) of a VDO volume.
    - mkfs.<fs_type> <vdo_device>: Format a VDO volume with a file system.
    - mount <vdo_device> <mount_point>: Mount the file system on the VDO volume.

    - vim /etc/fstab: Configure persistent mounts for VDO volumes.
    - df -h: Check disk space usage (useful for observing VDO efficiency).
    - rpm -q vdo, yum list installed vdo: Verify VDO package installation.
    - ssh, sudo -i: Used to connect to remote servers and gain root privileges.
    - mount -o remount,rw /: Example command shown.
    - lab advstorage-stratis start/finish: Lab commands for guided exercise on Stratis.
    - lab advstorage-vdo start/finish: Lab commands for guided exercise on VDO.
    - lab advstorage-review start/grade/finish: Lab commands for the chapter lab.
- Applications:

    - Deploying flexible and space-efficient storage solutions using Stratis for dynamically growing file systems and snapshots.
    - Optimizing storage utilization for data that contains duplicates or is compressible using VDO.
    - Reducing storage costs through deduplication and compression.

Chapter Title: Accessing Network-Attached Storage

- Key Concepts:

    - Accessing network-attached storage using the NFS (Network File System) protocol.
    - Identifying NFS share information from a server.

    - Mounting and unmounting NFS exports (shares) from the command line and at boot time.
    - Creating a directory to serve as a mount point for the NFS share.
    - Configuring persistent mounts for NFS shares using the /etc/fstab file.
    - Using the automounter (autofs service) to automatically mount an NFS file system on demand and unmount it when no longer in use. This provides benefits over persistent mounting like mounting only when accessed.
    - Automounter maps:
        - Indirect maps: Mount points are subdirectories within a base directory.
        - Direct maps: The map name specifies the full path to the mount point.
    - Automounter mapping files (/etc/auto.name or files in /etc/auto.master.d/) define mount points, mount options, and source locations. The format is mount point [options] source.
    - Mount options start with a dash (-) and are comma-separated (e.g., -rw,sync). fstype=<type> (e.g., nfs4) and strict are useful automounter options.
    - The nfsconf tool can be used to get, set, or unset NFS configuration parameters, including configuring an NFS client to use NFSv4.
    - NFS configuration file: /etc/nfs.conf. It uses sections ([keyword]) and key-value pairs (e.g., vers4.2=y). Lines starting with # or ; are ignored.
- Important Commands:

    - showmount -e <server>: Identify available NFS shares on a server.

    - mkdir <mount_point>: Create a directory for the mount point.
    - mount -t nfs <server>:<remote_share> <mount_point>: Mount an NFS share from the command line. Options can be specified with -o.
    - vim /etc/fstab: Edit /etc/fstab to configure persistent NFS mounts.
    - umount <mount_point>: Unmount an NFS share.
    - nfsconf --set <section> <key> <value>: Set an NFS configuration parameter using the nfsconf tool.
    - nfsconf --get <section> <key>: Get an NFS configuration parameter.
    - nfsconf --list: List all configuration parameters.
    - systemctl status autofs: Check the status of the automounter service.
    - systemctl enable --now autofs: Enable and start the automounter service.
    - vim /etc/auto.master.d/<map_file>: Configure the master map for the automounter.
    - vim /etc/auto.<name>: Create/edit automounter mapping files.
    - ssh, sudo: Used for remote access and elevated privileges.
    - yum install autofs: Install the automounter package.
    - lab netstorage-nfs start/finish: Lab commands for guided exercise on mounting NFS.
    - lab netstorage-autofs start/finish: Lab commands for guided exercise on automounting.
    - lab netstorage-review start/grade/finish: Lab commands for the chapter lab.
- Applications:

    - Accessing shared filesystems hosted on other servers over the network.

- Configuring network storage to be automatically available at boot for persistent access.
- Implementing on-demand access to network storage using the automounter, reducing resource usage compared to persistent mounts.
- Configuring NFS client behavior, such as specifying the desired NFS protocol version (e.g., NFSv4).

Chapter Title: Controlling the Boot Process

- Key Concepts:

  - Managing the boot process to control services offered and to troubleshoot and repair problems.
  - Understanding the Red Hat Enterprise Linux boot process.
  - Systemd targets: Sets of systemd units started to reach a desired system state. Important targets include graphical.target (multi-user, graphical/text login) and multi-user.target (multi-user, text login).
  - Setting the default target used when booting.
  - Booting a system to a non-default target temporarily.
  - Resetting the root password: A crucial task for system administrators, especially when the current password is lost. This is typically done from the boot loader (GRUB2).
  - Accessing a maintenance shell from the boot loader using rd.break on the kernel command line. In this mode, the root file system is mounted read-only under /sysroot.

    - Accessing emergency mode using systemd.unit=emergency.target on the kernel command line. This mode can be used to diagnose and fix file-system issues. In emergency mode, the root file system is mounted in read/write mode.
    - Repairing file system issues at boot: Manually fixing configuration errors (/etc/fstab) or corrupt file systems that prevent booting.
- Important Commands:

    - systemctl reboot: Reboot the system.
    - systemctl poweroff: Power down the system.
    - systemctl halt: Bring the system down to a safe state before manual power off.
    - systemctl get-default: Query the default target.
    - systemctl set-default <target_name>.target: Set the default target persistently.
    - systemctl isolate <target_name>.target: Switch to a new target at runtime.
    - parted /dev/<disk> print: Used in guided exercise to confirm partitioning.
    - mount: Show mounted file systems and their options.
    - sulogin: Used to log in as root for maintenance.
    - Editing the GRUB2 boot entry: Access by pressing e at the boot menu, add parameters like rd.break or systemd.unit=emergency.target to the linux line, and boot with Ctrl+x.
    - chroot /sysroot: Change the root directory to /sysroot when using rd.break.

    - mount -o remount,rw /sysroot: Remount /sysroot as read/write in rd.break mode.
    - passwd root: Change the root password.
    - autorelabel: Trigger SELinux relabeling if needed after password reset.
    - exit: Exit a shell, including the maintenance shell or emergency mode.
    - Ctrl+d: Continue the boot process from emergency mode or after exiting a shell in maintenance mode.
    - fsck <device>: Check and repair file systems (implied in troubleshooting, not explicitly detailed).
    - vim /etc/fstab: Edit the file system table to fix errors.
    - lab boot-selecting start/finish: Lab commands for guided exercise on boot targets.
    - lab boot-resetting start/finish: Lab commands for guided exercise on password reset.
    - lab boot-review start/grade/finish: Lab commands for the chapter lab.
    - lab rhcsa-compreview1 break1: Lab command to simulate a boot failure for review.
- Applications:

    - Managing system shutdown and restart.
    - Switching between different runlevels or system states (targets) at runtime.
    - Configuring the system to boot into a specific target by default.
    - Recovering a lost root password.

- Diagnosing and fixing issues that prevent the system from booting correctly, such as file system errors or /etc/fstab misconfigurations, using emergency or rescue mode.

Chapter Title: Managing Network Security

- Key Concepts:
  - Controlling network connections to services using the system firewall and SELinux rules.
  - The netfilter subsystem allows kernel modules to inspect every network packet.
  - Firewalld: Simplified firewall management by classifying network traffic into zones. Each zone has its own list of ports and services. The public zone is set as the default zone.
  - Firewalld services are pre-defined rules for common applications.
  - Accepting or rejecting network connections to system services using firewalld rules.
  - SELinux Port Labeling: Network traffic is tightly controlled by the SELinux policy. Network ports are labeled (e.g., port 22/TCP has label ssh_port_t).
  - SELinux checks if a process's label is allowed to bind a specific port label when the process tries to listen on that port.
  - Controlling whether network services can use specific networking ports by managing SELinux port labels.
- Important Commands:

- Firewalld Tools (firewall-cmd):
  - firewall-cmd --get-services: List pre-defined services.
  - firewall-cmd --zone=<zone> --add-port=<port>/<protocol>: Add a port to a zone.
  - firewall-cmd --zone=<zone> --remove-port=<port>/<protocol>: Remove a port from a zone.
  - firewall-cmd --zone=<zone> --add-service=<service>: Add a service to a zone.
  - firewall-cmd --zone=<zone> --remove-service=<service>: Remove a service from a zone.
  - --permanent: Make the change persistent across reboots.
  - --runtime: Apply the change immediately (default).
  - firewall-cmd --reload: Reload firewalld rules after permanent changes.
- SELinux Port Management (semanage port):
  - semanage port -l: List existing port labels.
  - semanage port -a -t <port_label> -p tcp|udp <PORTNUMBER>: Add a port to an existing label.
  - semanage port -d -t <port_label> -p tcp|udp <PORTNUMBER>: Delete a port label record.
  - semanage -C: View local changes to the default policy.
- ssh, sudo: Used for remote access and elevated privileges.
- lab netsecurity-firewalls start/finish: Lab commands for guided exercise on firewalls.
- lab netsecurity-ports start/finish: Lab commands for guided exercise on SELinux port labeling.

    - lab netsecurity-review start/grade/finish: Lab commands for the chapter lab.
- Applications:

    - Configuring the system firewall to allow or deny network access to specific ports or services.
    - Managing network security using firewalld zones.
    - Ensuring that network services can bind to their required ports according to the SELinux policy.
    - Troubleshooting network access issues related to firewalld or SELinux port labeling.

Chapter Title: Installing Red Hat Enterprise Linux

- Key Concepts:

    - Installing Red Hat Enterprise Linux on servers and virtual machines.
    - The Anaconda installer is the standard RHEL installation program.
    - Key installation configuration items in Anaconda: Installation Source, Software Selection (e.g., Minimal Install), Installation Destination (disk selection and partitioning), Root Password, User Creation, Network Configuration, Time & Date, Language & Keyboard.
    - Understanding partitioning schemes (MBR, GPT) and file system types is needed for selecting the Installation Destination.
    - Automating the installation process using Kickstart.

- Kickstart file: A text file containing configuration directives that automate the Anaconda installer.
- Kickstart file syntax: Starts with #!, uses directives (e.g., lang, keyboard, timezone, rootpw, part, clearpart, autopart), and sections (e.g., %packages, %post) marked with % and %end. Comments start with #.
- Kickstart directives include those for system configuration (lang, keyboard, timezone, authselect, rootpw, selinux, firstboot, skipx, services), disk partitioning (part, clearpart, autopart), and scripting (%pre, %post, %onerror, %onboot).
- The %packages section specifies software to install, listing individual packages, package groups (@), or environment groups (@^).
- A Kickstart file needs to be made available to the installer, commonly via HTTP (inst.ks=URL boot option).
- Validating the syntax of a Kickstart file is important before use.
- Installing and configuring Virtual Machines (VMs) on a RHEL server.
- The virt Yum module provides packages to make RHEL a virtualization host.
- The Cockpit web console can be used to install and manage virtual machines after installing the cockpit-machines package.

- Important Commands:

  - Kickstart-related:
    - vim <kickstart_file.cfg>: Create/edit a kickstart file.
    - ksvalidator <kickstart_file.cfg>: Validate the syntax of a kickstart file.

        - inst.ks=http://<server>/path/to/kickstart.cfg: Kernel boot option used to specify the location of a kickstart file for automated installation.
        - sudo cp <kickstart_file.cfg> /var/www/html/ks-config/: Example of making a kickstart file available via HTTP.
    - Virtualization-related:
        - virt-host-validate: Validate host readiness for virtualization.
        - virsh: Command-line interface for managing VMs.
        - yum module enable virt:rhel (Implied by virt module concept)
        - yum install cockpit-machines (Implied by Cockpit concept).
        - Cockpit Web Console: Graphical interface for VM management.
    - General:
        - ssh, sudo: Used for remote access and elevated privileges.
    - lab installing-kickstart start/finish: Lab commands for guided exercise on Kickstart.
    - lab installing-review start/grade/finish: Lab commands for the chapter lab (includes Kickstart installation).
- Applications:

    - Performing standard interactive installations of RHEL on physical servers.
    - Automating large-scale RHEL deployments using Kickstart for consistent and repeatable installations.
    - Creating and configuring virtual machines on a RHEL server for consolidation or testing.
    - Using graphical tools like Cockpit for simpler virtual machine management.

Chapter Title: Comprehensive Review

- Key Concepts:
    - Reviewing and refreshing knowledge and skills learned throughout Red Hat System Administration II.
    - This chapter consolidates tasks and concepts covered in previous chapters.
    - Areas covered include: Improving Command-line Productivity, Scheduling Future Tasks, Tuning System Performance, Controlling Access to Files with ACLs, Managing SELinux Security, Managing Basic Storage, Managing Logical Volumes, Implementing Advanced Storage Features, Accessing Network-Attached Storage, Controlling the Boot Process.
    - Specific review tasks involve troubleshooting boot failures, maintaining servers, configuring/managing file systems and storage, and configuring/managing server security.
- Important Commands:
    - This chapter utilizes commands from all previous chapters, including but not limited to:
        - Boot troubleshooting/maintenance: systemctl commands (reboot, poweroff, get-default, set-default), grubby, using boot loader options (rd.break, systemd.unit=emergency.target), mount, sulogin, passwd.

        - Storage management: parted, mkfs.xfs, mkswap, swapon, swapoff, LVM tools (pvcreate, vgcreate, lvcreate, etc.), vim /etc/fstab, mount, systemd-tmpfiles.
        - Network storage: mount, umount, vim /etc/fstab, autofs configuration files/commands, nfsconf.
        - Security: getfacl, setfacl, firewall-cmd, getenforce, setenforce, semanage, sealert, ssh-keygen.
        - Command-line/scripting: grep, shell scripting constructs (loops, conditionals), ssh.
    - lab rhcsa-compreview1 start/break1/grade/finish: Lab commands for the first comprehensive review lab (boot/maintenance).
    - lab rhcsa-compreview2 start/grade/finish: Lab commands for the second comprehensive review lab (storage).
    - lab rhcsa-compreview3 start/grade/finish: Lab commands for the third comprehensive review lab (security).
- Applications:

    - Applying the skills learned in the course to troubleshoot and solve complex system administration problems.
    - Performing integrated tasks involving multiple system areas (e.g., storage and security, boot issues and maintenance).
    - Demonstrating proficiency in key RHEL system administration tasks required for practical exams like the RHCSA.
