Advanced Unix Commands

Kameswari Chebrolu

Outline

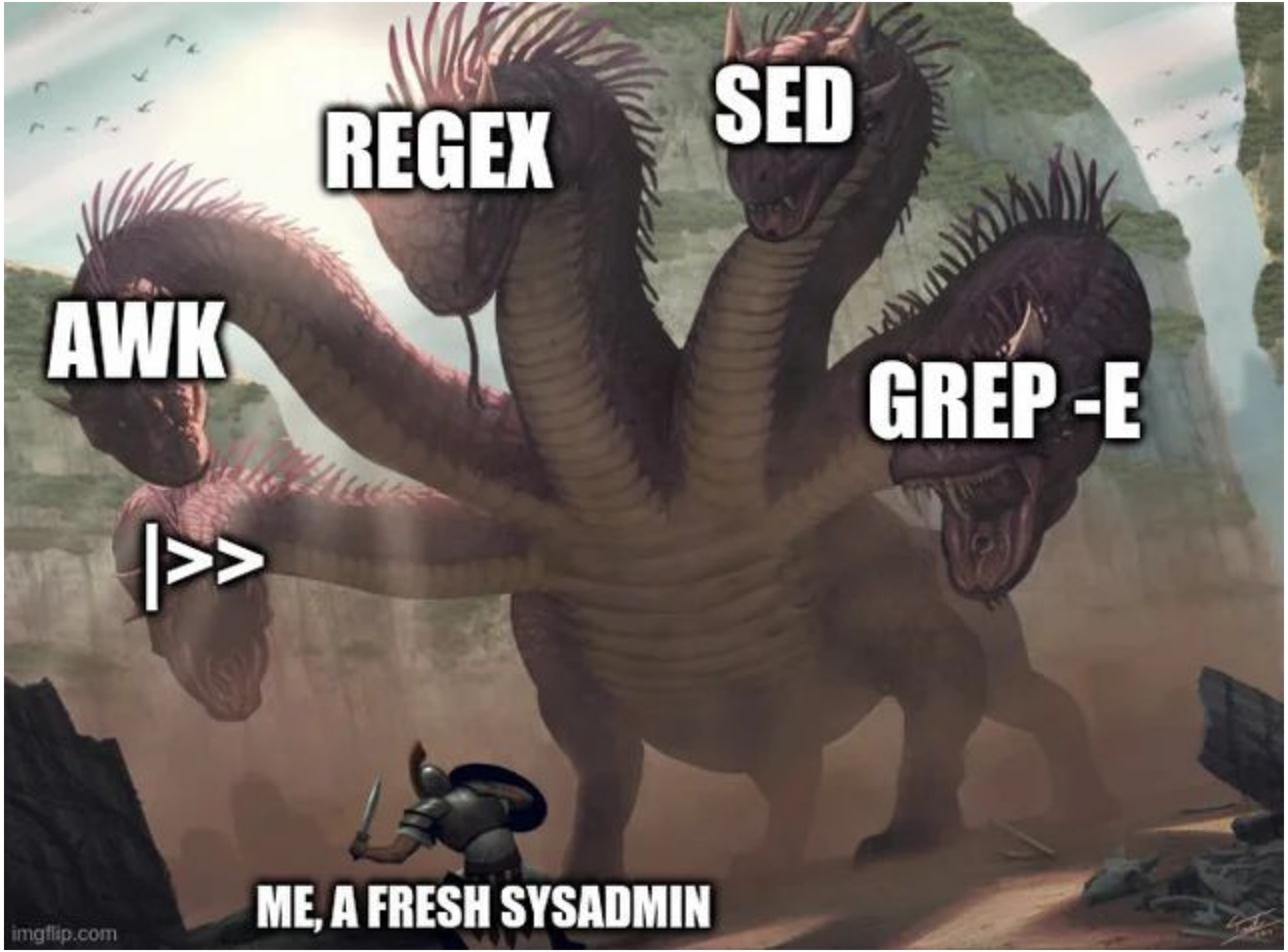
- ~~File and Directory Commands~~
- ~~File Viewing and Editing Commands~~
- Commands for File Analysis
- Process Management
- Security and Permissions

File Analysis Commands

Commands

- wc
- regex
- grep
- find
- cut

- paste
- sort
- uniq
- zip/tar
- redirection (>, >>, <)
- Pipe (|)



https://preview.redd.it/yjtwtofkxgy51.jpg?width=640&crop=smart&auto=webp&s=166b65dac9fb037c6d569744d12adbd3d84491ea

wc

- wc’s motto: Every word counts!
- Counts the number of lines, words, and characters in a file or input from standard input
  - Will tell you if your file is too long, too short, or just right :-)
- Use Case:
  - Quickly obtaining statistics about text files
  - Often combined with other commands using pipes to process and analyze text
- Syntax : wc [OPTION] [FILES]
  - [FILES]: File(s) you want to analyze
    - If no file is provided, wc reads from standard input

- Output of wc typically consists of three numbers (when no specific option is used)
  - Number of Lines: Total number of lines in the file
  - Number of Words: Total number of words
  - Number of Bytes: Total size of the file in bytes
- Key Options
  - -l: Count lines
  - -w: Count words
  - -c: Count bytes
  - -m: Count characters
  - -L: Print the length of the longest line (in characters)

Demo

wc

WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.
OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!
BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!
IT'S HOPELESS!
EVERYBODY STAND BACK.
I KNOW REGULAR EXPRESSIONS.
PERL!
TAP TAP

https://xkcd.com/208/

Regular Expressions (regex)

- regex: a pattern that matches a set of strings
  - Used in text editors, programming languages, and command-line tools
- <u>Metacharacters:</u> characters with special meaning
  - “^” beginning of a line (Can also mean “not” if inside [])
  - “$” end of line
  - “.” match any single character
  - “\” escape a special character
  - “|” or operation i.e. match a particular character set on either side

Quantifiers: specifying the number of occurrences of a character

- “*” Match the preceding item zero or more times
- “?” Match the preceding item zero or one time
- “+” Match the preceding item one or more times
- “{n}” Match the preceding item exactly n times
- “{n,} Match the preceding item at least n times
- “{,m}”Match the preceding item at most m times
- “{n,m}” Match the preceding item from n to m times

Groups and Ranges

- “ ( )” group patterns together
- “{   }” match a particular number of occurrences (seen before)
- “[ ]” match any character from a range of characters
  - ab[xyz]c "abxc" and "abyc"and "abzc"
  - [^…..] matches a character which is not defined in the square bracket
  - [a-z] matches letters of a small case from a to z
  - [A-Z] matches letters of an upper case from A to Z
  - [0-9] matches a digit from 0 to 9.

grep

- Grep: Global Regular Expression Print
- Searches for specific patterns within files or input provided via standard input
  - Used for text searching and processing
- Syntax : grep [OPTIONS] PATTERN [FILE...]
  - [OPTIONS]: Optional flags modify the behavior of grep
  - PATTERN: The regular expression pattern to search for
  - [FILE]: One or more files to search
    - If no file is specified, grep reads from standard input

- Key Options
  - -i: Ignore case (case-insensitive search)
  - -v: Invert match (show lines that do not match the pattern)
  - -r or -R: Recursively search directories
  - -n: Show line numbers with matching lines
  - -c: Count the number of matching line

- -H: Print the filename for each match
  - Useful when searching multiple files
- -o: Print only the matched parts of a line
- -E: Use extended regular expressions
- -w: match only whole words
- -A: Displays lines of text that appear after the matching line
- -B: Displays lines of text that appear before the matching line
- -C: Displays lines of text that appear both before and after the matching line

Demo

grep

find

- Used to search for files and directories based on various criteria
  - Can search for files by name, size, type
  - Can perform actions (execute commands) on found files
- Use case: Locate specific files, clean up old files, or performing actions on files that match certain conditions

- find [PATH] [OPTIONS] [CRITERIA] [ACTIONS]
  - [PATH]: The directory or directories to start the search from (default is the current directory)
  - [OPTIONS]: Optional flags that modify the behavior of find
  - [CRITERIA]: Conditions used to match files (e.g., by name, size, type)
  - [ACTIONS]: Actions to perform on the matched files (e.g., print, delete)

- Key Options and Criteria
  - -name: Search for files by name
  - -iname: Case-insensitive search for files by name
  - -type: Search for files by type
    - f: Regular file
    - d: Directory
  - -size: Search for files by size
    - +: Larger than
    - -: Smaller than
    - c: Size in bytes.

- -perm: Search for files or directories based on their permissions
- -mtime: Search for files based on modification time
  - +: More than n days ago
  - -: Less than n days ago
  - n: Exactly n days ago
- -exec: Execute a command on each found file
  - -delete: Delete files that match the search criteria
  - -print: Print the path of each found file (default action)

Demo

find

cut

- Used to extract specific sections of text from each line of input data
  - Useful for processing and filtering columns of data from text files, logs, or command output
  - Effective with structured data, such as CSV files or delimited text,
- Syntax: cut [OPTIONS] [FILE...]
  - FILE...: The file(s) to process
    - If no file is specified, cut reads from standard input

- Key Options
  - -f: Specifies the fields to be extracted
    - Fields are separated by a delimiter (tab is default)
  - -d: Defines the delimiter that separates fields in the input data
    - Default behavior: use the input delimiter as the output delimiter
  - -c: Extracts specific characters from each line of the input
  - -b: Extracts specific bytes from each line of input
  - --complement: Complement the selection
    - Displays all bytes, characters, or fields except the selected
  - --output-delimiter: Allows to specify a different output delimiter string

Demo

cut

paste

- Used to merge lines of files horizontally, creating columns of data
  - Combines corresponding lines from each file specified as arguments, separating them by a delimiter (which defaults to a tab)
- Use case:
  - Useful for joining data from multiple files or streams
    - Creates side-by-side comparisons or concatenated outputs
  - cat command merges files vertically (one after the other)
  - paste merges files horizontally, placing lines from different files side by side

- Syntax: paste [OPTIONS] [FILE...]
  - FILE...: The files to be merged
    - If no files are specified, paste reads from standard input
- Key Options
  - -d: Specifies a custom delimiter to use between merged lines
  - -s: Merges lines from one file sequentially, rather than in parallel with other files.
  - -: Indicates that standard input should be used in place of a file.

Demo

paste

sort

- Used to arrange lines of text files or input data in a specific order
  - By default, sorts lines alphabetically or numerically based on the first character, but can be customized
- Use case: Organize data for better readability, prepare data for further processing
- Syntax : sort [OPTIONS] [FILE...]
  - FILE...: The files to be sorted
    - If no files are specified, sort reads from standard input

- Key Options
  - -n: Sorts numerically, treating the first part of each line as a number
    - Useful for sorting lists of numbers or data that includes numeric fields.
  - -r: Reverses the sort order
  - -k: Sorts based on a specific field within each line
  - -t: Specifies a custom delimiter that separates fields in the input.
  - -u: Removes duplicate lines from the output, showing only unique entries
  - -M: Sorts lines based on the first three characters of the month name

Demo

sort

uniq

- Used to filter out or report repeated lines in a file or input data
  - Only works on adjacent lines
    - Identifies or removes duplicates that are directly next to each other
  - Does not perform any fuzzy matching
  - Only identifies lines that are exactly the same
- Use case: Commonly used in combination with sort to process sorted data

- Syntax : uniq [OPTIONS] [INPUT] [OUTPUT]
  - INPUT: The file to be processed
    - If no input file is specified, uniq reads from standard input
  - OUTPUT: The file where the results will be written
    - If no output file is specified, results are written to standard output

- Key Options
  - -c: Prefixes each line with the number of times it appears in the input
    - Useful for counting occurrences of each line
  - -d: Displays only the lines that are repeated (duplicates)
  - -u: Displays only the lines that are unique, excluding all repeated lines
  - -i: Ignores case when comparing lines
  - -f: Skips a specified number of fields before performing comparisons
  - -s: Skips a specified number of characters before performing comparisons.

zip

- Used to create compressed archive files
  - Bundles multiple files and directories into a single .zip file
  - Supports both compression and file management tasks
    - File management: can add, update, and delete files within an archive
  - .zip files are supported on many operating systems
- Use case: Helps create backups, package and distribute files
- Syntax: zip [OPTIONS] ARCHIVE FILES…
  - ARCHIVE: The name of the output .zip file to create or update
  - FILES...: The files and directories to include in the archive.

- Key Options
  - -r: Recursively include directories and their contents
  - -u: Update an existing archive with new or changed files
  - -d: Delete files from an existing archive
  - -x: Exclude files or directories from the archive
  - -e: Encrypt the archive with a password
  - -s : splits a large archive into multiple smaller files

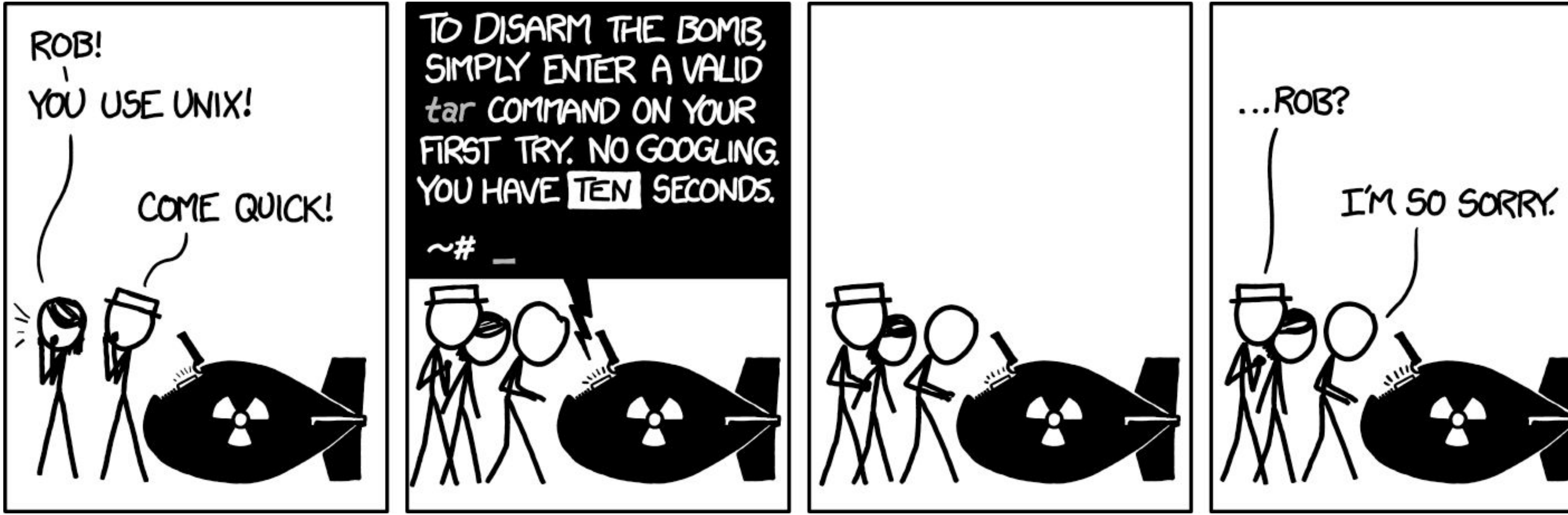
- Key Options
  - -d: Specify the directory to extract files into
  - -l: List the contents of the archive without extracting
  - -o: Overwrite existing files without prompting
  - -n: Never overwrite existing files
  - -x: Exclude specific files from extraction.

tar

- Used to create and manipulate archive files
  - Can bundle multiple files and directories into a single archive file, often with a .tar extension
  - Can extract files from archives
  - Supports various compression methods to reduce the archive using gzip, bzip2, or xz
  - Preserves file metadata such as permissions, ownership, and timestamps
    - Very useful for backups and transfers

- Syntax : tar [OPTIONS] [ARCHIVE] [FILES...]
  - ARCHIVE: The name of the archive file to create, extract, or manipulate
  - FILES...: The files and directories to include in the archive or extract from it.

- Key Options
  - -c: Create a new archive
  - -x: Extract files from an archive
  - -t: List the contents of an archive without extracting
  - -f: Specifies the archive file name
    - Must be followed by the name of the archive file
  - -z: Compress or decompress using gzip
  - -j: Compress or decompress using bzip2
  - -J: Compress or decompress using xz
  - -v: Verbose mode, displays the progress of the operation


ROB!
YOU USE UNIX!
COME QUICK!
TO DISARM THE BOMB, SIMPLY ENTER A VALID tar COMMAND ON YOUR FIRST TRY. NO GOOGLING. YOU HAVE TEN SECONDS.
~# _
...ROB?
I'M SO SORRY.

https://xkcd.com/1168/

- Zip:
  - Combines both compression and archiving in a single step
  - Often used for cross-platform compatibility
- Tar:
  - Primarily used for bundling files, with compression applied separately
  - Favored for its efficiency and detailed preservation of file attributes

Input/Output

- A process is an instance of a program that is being executed
- Stream: a special file that either continuously receives text in or pushes text out
- When you run a command, OS creates a process to execute that command

- Whenever a process starts, process is given access to three “standard” streams
    - fd is file descriptor
    - Standard input (stdin; fd is 0)
    - Standard output (stdout, fd is 1)
    - Standard error (stderr, fd is 2); used when an error has occurred
- When a process starts
  - stdout and stderr are configured to print whatever they receive to the terminal screen
  - stdin is configured to read input from the user’s keyboard

Redirection

- Redirection controls where the output of a command goes and where input comes from
  - Allows you to redirect standard input (stdin), standard output (stdout), and standard error (stderr)
- > (Output Redirection) : Redirects the standard output (stdout) of a command to a file
  - If the file already exists, it will be overwritten
  - Syntax: command > file

- >> (Append Redirection) : Redirects the standard output (stdout) of a command to a file, but instead of overwriting, appends output end of the file
  - Syntax: command >> file
- < (Input Redirection): Redirects the standard input (stdin) for a command from a file
  - Instead of typing input directly into the command, it reads the input from the specified file
  - Syntax: command < file

- command > file same as command 1> file (stdout redirected, stderr still screen)
- command 2> file (send stderr to file, stdout is screen)
- command 2> error.txt 1> out.txt (send both to different files)
- command > file 2>&1 (send both to same file)
- command 2> /dev/null (suppress error messages)
  - /dev/null is a special file that discards anything written to it

pipe

- How many files and folders in /etc ?
  - ls /etc > temp.txt; wc -l temp.txt; rm temp.txt
- A pipe (|) allows the output of one command to be used as input for another command
  - Shell connects the stdout of the first command directly to the stdin of the second command
    - Operates entirely in memory
    - Unidirectional: flows from left to right

  - Enables chaining of multiple commands together to perform complex operations
- Use case: Process data through a series of commands without needing to store intermediate results in temporary files
- Syntax: command1 | command2 | command3 ...

Command Substitution

- How to use the output of a command in the arguments of another
  - Note: pipes are great for sharing stdin and stdout between processes
- Command substitution: Output of a command replaces the command itself
  - $(command) or
  - `command`

Outline

- ~~File and Directory Commands~~
- ~~File Viewing and Editing Commands~~
- ~~Commands for File Analysis~~
- Process Management
- Security and Permissions

Process Management

Commands

- ps
- pkill

Process Management

- Methods and tools to control, monitor, and interact with processes running on the system
- A process is a running instance of a program
  - Each process in Linux has a unique Process ID (PID)
  - From a process, another process can be created
    - Achieved via fork system call

- Parent-child relationship exists between the two processes
  - PID (Process ID): A unique identifier assigned to each process.
  - PPID (Parent Process ID): The PID of the process that started (or "parented") the current process.
- init has process id 1
  - Parent of all processes
  - Executed by the kernel during the booting of a system

- Process States:
  - Running: The process is currently executing
  - Sleeping: The process is waiting for an event (e.g., I/O completion)
  - Stopped: The process has been stopped, usually by receiving a signal
  - Zombie: The process has completed execution but still has an entry in the process table

ps

- Displays information about the currently running processes on the system
  - Default output is a list of the processes associated with the command-line
  - Shows unique process id (pid), terminal used, amount of CPU time, and the program name
- Use case: Commonly used to monitor running processes, check for specific processes, or troubleshoot system performance issues
- Syntax: ps [options]

- Key Options
  - -e or -A: Lists all processes running on the system
  - -f: Displays full-format listing
  - -u username: Shows processes owned by the specified user
  - -p PID: Displays information about the specified process ID(s)
  - -C command_name: Filters processes by the command name
  - aux: Displays detailed information about all processes
    - “a”: display the processes of all users
    - “u”: user-oriented format → more details
    - “x”: list the processes without a controlling terminal
      - Started on boot time and running in the background

- PID: Process ID - The unique identifier for the process.
- TTY: Terminal type associated with the process.
- TIME: Total CPU time the process has consumed.
- COMMAND: The command that started the process.
- USER: The user who owns the process (shown with options like aux).
- %CPU: The percentage of CPU usage (shown with options like aux).
- %MEM: The percentage of memory usage (shown with options like aux).
- VSZ: Virtual memory size (shown with options like aux).
- RSS: Resident Set Size, the non-swapped physical memory that the task has used (shown with options like aux).
- STAT: Process state (e.g., R for running, S for sleeping) (shown with options like aux).
- START: The start time of the process

pkill

- Used to send signals to processes to request actions like termination, suspension, or restarting
  - Targets processes based on their names or other attributes, rather than process ID (PID)
    - Kill another command which is similar but needs PIDs
- Use Case: Control processes by sending them specific signals
  - Request actions like termination, suspension, or restarting
- Syntax: pkill [options] pattern

- Key Options
  - -s SIGNAL: Specifies the signal to send to the matching processes
    - If omitted, the default signal is SIGTERM
    - To know what SIGNALs available, use kill -l
  - -f: Matches against the full command line of the processes, not just the process name
  - -u USER: Targets processes owned by the specified user
  - -t TTY: Targets processes associated with the specified terminal
  - -P PID: Targets child processes of the specified parent PID

How Windows ask a process to terminate
How Linux ask a process to terminate

https://i.redd.it/2ndv691za33a1.png

Outline

- ~~File and Directory Commands~~
- ~~File Viewing and Editing Commands~~
- ~~Commands for File Analysis~~
- ~~Process Management~~
- Security and Permissions

Security and Permissions

Commands

- su
- sudo
- Access control

Superuser

- Superuser: user with super powers
  - A real user account (often root) that can do just about anything (modify/delete files, run any programs etc)
- For security reasons, “su” was introduced
  - Can mean ‘superuser’ or ‘switch user’
  - Helps change to another user without having to log out and in
    - Terminal session switched to the other user
    - Requires password of the other user
  - Administrators spend most time using normal account, when needed switch to superuser, do task and logout

- Syntax: su [options] [username]
    - username: User you want to switch to
        - If not specified, command switches to root
- Key Options:
    - -c [command]: Executes a single command as the specified user and returns to previous user after command is run
    - -s [shell]: Specifies which shell to use when switching users
    - -p or --preserve-environment: Preserves the current environment variables instead of loading the new user's environment

- Further improvement, “sudo” was introduced
  - “switch user and do this command”
    - Does not fully switch to that user’s environment
  - Asks for the current user's password (not root's)
    - Permissions for using sudo are defined in the /etc/sudoers file
      - Administrators can specify which users are allowed to run which command
    - Runs the command with elevated privileges
    - Entered password is cached for a default period (usually 15 minutes)
      - No need to re-enter it for subsequent sudo commands

  - Prevents long-lived terminal sessions with dangerous powers
- Use Case: Grants users temporary access to perform administrative tasks
  - Tasks otherwise restricted to root user or a system administrator
- Be very careful when using sudo or su

- Key Options:
  - -u [user]: Runs the command as a specified user, instead of root
  - -l: Lists the commands that the current user is allowed to run with sudo
  - -k: Invalidates the current user’s cached credentials, forcing sudo to prompt for a password again.
  - --preserve-env or -E: Preserves the current environment variables when running a command

- Why Use sudo Instead of su?
  - More Secure: Unlike su, sudo doesn’t require sharing the root password, which limits security risks
  - Granular Control: Administrators can limit which commands users can run with sudo
  - Auditability: Actions performed with sudo can be logged, helping track which users executed which commands

Access Control

- UNIX is a multi-user system
- Every file and directory (in your account) can be protected from or made accessible to other users. How?
- Permissions for a file or directory may be any or all of
  - r - read; w - write; x - execute
  - a directory must have both r and x permissions if the files it contains are to be accessed

- Each permission (rwx) can be controlled at three levels:
    - u (user = yourself)
    - g (group, a set of users)
    - o (others, everyone else)
- File access permissions are displayed using “ ls -l”
- Use Case: Sensitive information is protected and only accessible to authorized personnel

- First field: - for File, d for Directory, l for Link
- Second,third,fourth fields: permissions for owner, group and others
- Fifth field: specifies the number of links or directories inside this directory
- Sixth field: user
- Seventh field: group
- Eighth field: size in bytes (use -lh option for better understanding)
- Ninth field: date of last modification
- Tenth field: name of the file/directory

chmod

- “chmod” command helps change access permissions for files you own
  - chmod [OPTIONS] MODE FILE(s)/directory(s)
  - Only root, the file owner or user with sudo privileges can change the permissions of a file
- Use Case: Set appropriate permissions on files and directories of web server to ensure both security and functionality

- Symbolic Mode:
  - u - File owner.
  - g - Users who are members of the group.
  - o - All other users.
  - a - All users, identical to ugo
  - - Removes the specified permissions.
  - + Adds specified permissions
  - = Changes the current permissions to the specified permissions
    - If no permissions are specified after the = symbol, all permissions from the specified user class are removed
  - E.g. chmod g=r filename; chmod a-x filename;

- Numeric mode:
  - r (read) = 4
  - w (write) = 2
  - x (execute) = 1
  - no permissions = 0
  - chmod 640 file
    - octal notation, user has r+w, group has read, others have none
  - chmod -R 700 dirname
    - Recursively set read, write, and execute permissions to the file owner
    - No permissions for all other users on a given directory

References

- https://ubuntu.com/tutorials/command-line-for-beginners#1-overview
- https://linuxize.com/ (good resource, use search box for info on different commands!)