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

<u>Quantifiers:</u> 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.

- What did the paste command say to cut during their collaboration?
  - "You divide, and I conquer!"

Demo

paste

References

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