Question 1: Current Shell Process ID

Answer: $$

Explanation: $$ returns the process ID of the current shell.

Question 2: Match a String at the Beginning of a Line

Answer: ^Confident

Explanation: ^ anchors the regex to the start of the line.

Question 3: Regular Expression with Brackets

Answer: , 135, 1155, 915, 513, 18.8

Explanation: The pattern matches four groups of numbers followed by one decimal number.

Question 4: Remove Permissions for User & Group

Answer: chmod ug -rwx software

Explanation: chmod ug -rwx removes read, write, and execute permissions for user and group.

Question 5: Power of UNIX Commands with Piping

Answer:

grep "CSCE3600" grades.txt: Finds all lines containing "CSCE3600"

sort -k 2,2gr: Sorts lines by the second column in descending order.

uniq: Removes duplicate lines.

Question 6: Output Redirection and Append

Answer: > and >>

Explanation: > redirects output to a file, >> appends to a file.

Question 7: Command Matching

head -n5 send.c: Prints the first 5 lines of send.c.

wc -l send.c: Counts the number of lines in send.c.

whatis sort: Displays a brief description of the sort command.

Question 8: Sed Command

Answer: sed -n '3,6p' lines

Explanation: sed -n prints lines 3 to 6 from the file.

Question 9: Regex Matching

/ab+c/: Matches "ab" followed by one or more "b" characters and "c".

Question 10: Display Logged-in Users

Answer: who

Explanation: who shows users logged in on the local system.

Question 11: File Comparison Utility

Answer: diff

Explanation: diff compares files and shows the differences.

Question 12: rm Command Prompt Option

Answer: -i

Explanation: rm -i prompts for confirmation before deleting files.

Question 13: ls vs cat

Answer: name, contents

Explanation: ls shows the name of files, cat shows their contents.

Question 14: Command Line Options

Answer: options

Explanation: Options modify the behavior of commands.

Question 15: Shell Name Display

Answer: True

Explanation: echo $0 shows the name of the shell you're using.

Question 16: cp Overwrites Existing Files

Answer: True

Explanation: cp overwrites the destination file without warning.

Question 17: Locate Executable Files

Answer: whereis

Explanation: whereis locates binary files, manuals, and source code.

Question 18: Environment Variables

Answer: export

Explanation: export makes shell variables available to child processes.

Question 19: Linux OS Type

Answer: multiuser

Explanation: Linux supports multiple users simultaneously.

Question 20: Octal Permissions

Answer: 640

Explanation: 640 grants read/write permissions to the owner, read-only to the group, and no permissions to others.

Question 21: Accessing OS Services

Answer: system calls

Explanation: System calls provide access to OS services.

Question 22: Multitasking and PIDs

Answer: True

Explanation: In multitasking, each process has a unique process ID (PID).

Question 23: Command Interpreter

Answer: shell

Explanation: The shell interprets and executes user commands.

Question 24: Filesystem read System Call

Answer: True

Explanation: read is a system call for reading from file descriptors.

Question 25: System Call Expense

Answer: True

Explanation: System calls are expensive due to context switching.

Question 26: Focus of C vs. C++

Answer: False

Explanation: C focuses more on processes; C++ focuses more on data.

Question 27: Home Directory Symbol

Answer: ~

Explanation: ~ refers to the user's home directory.

Question 28: Checking Server Connection

Answer: ping

Explanation: ping tests connectivity to a server.

Question 29: Regular Expression Matching

Answer: True

Explanation: Regex matches the longest string first.

Question 30: Match a Line with One Character

Answer: ^.$

Explanation: ^.$ matches exactly one character in a line.

Question 31: gawk Buffers

Answer: fields and records

Explanation: gawk processes data in fields and records.

Question 32: Print Literal $var

Answer: echo \$var

Explanation: \$ escapes the dollar sign to print it literally.

Question 33: Bash Script Execution

Answer: by the shell command interpreter

Explanation: Bash scripts are executed by the shell interpreter.

Question 34: Bash String Substring

Answer: ijkl

Explanation: ${str:8:3} extracts 3 characters starting from the 8th.

Question 35: gawk Special Variable

Answer: True

Explanation: $0 in gawk refers to the entire line.

Question 36: Bash Script Output

Answer: num1 is not greater than num2 num1 is not equal to num2

Explanation: The conditions in the script produce this output.

Question 37: Purpose of test Keyword

Answer: The test keyword is used to check if the file "example.txt" exists and is not empty.

Explanation: test -s checks if a file exists and is not empty.

Question 38: String Comparison in Bash

Answer: if [ $m == $n ]

Explanation: Use [ $m == $n ] to compare two strings in Bash.

Basic Characters

. — Matches any character except a newline.

a — Matches the character a. Regular characters like this match themselves.

\ — Escape character. Escapes special characters, allowing them to be treated literally (e.g., \. matches a literal period).

Anchors

^ — Matches the start of the string or line (in multiline mode).

$ — Matches the end of the string or line (in multiline mode).

\A — Matches the start of the string (ignores multiline).

\Z — Matches the end of the string, or before a newline at the end.

\z — Matches the absolute end of the string.

\b — Word boundary (matches the position between a word character and a non-word character).

\B — Non-word boundary.

Character Classes

[abc] — Matches any single character inside the brackets (e.g., a, b, or c).

[^abc] — Matches any single character not inside the brackets.

[a-z] — Matches any character within the specified range (e.g., lowercase letters).

\d — Matches any digit (equivalent to [0-9]).

\D — Matches any non-digit character.

\w — Matches any word character (letters, digits, and underscores).

\W — Matches any non-word character.

\s — Matches any whitespace character (spaces, tabs, line breaks).

\S — Matches any non-whitespace character.

[[:alpha:]] — POSIX character class for letters.

[[:digit:]] — POSIX class for digits.

[[:alnum:]] — POSIX class for alphanumeric characters.

[[:punct:]] — POSIX class for punctuation characters.

[[:space:]] — POSIX class for whitespace characters.

Quantifiers

\* — Matches 0 or more of the preceding element.

+ — Matches 1 or more of the preceding element.

? — Matches 0 or 1 of the preceding element (makes it optional).

{n} — Matches exactly n occurrences of the preceding element.

{n,} — Matches n or more occurrences.

{n,m} — Matches between n and m occurrences.

Greedy vs. Lazy Quantifiers:

By default, quantifiers are greedy, meaning they try to match as much as possible.

Adding a ? after a quantifier makes it lazy, meaning it will match as little as possible.

.\* (greedy) vs. .\*? (lazy)

Grouping and Capturing

() — Captures the matched substring for later use. Each set of parentheses represents a capturing group.

(?:) — Non-capturing group. Groups expressions without capturing.

(?P<name>...) — Named capturing group.

\n — Backreference to the nth captured group (e.g., \1 for the first group).

Lookahead and Lookbehind

Lookahead:

(?=...) — Positive lookahead. Ensures that what follows matches ..., but doesn't consume the characters.

(?!...) — Negative lookahead. Ensures that what follows does not match ....

Lookbehind:

(?<=...) — Positive lookbehind. Ensures that what precedes matches ..., but doesn't consume the characters.

(?<!...) — Negative lookbehind. Ensures that what precedes does not match ....

Alternation (OR)

| — Alternation operator. Matches either the expression before or after the |. (e.g., a|b matches a or b).

Modifiers (Flags)

(?i) — Case-insensitive matching.

(?m) — Multiline mode (makes ^ and $ match the start/end of each line).

(?s) — Dot-all mode (makes . match newline characters).

(?x) — Free-spacing mode (ignores whitespace and allows comments in the pattern).

(?U) — Ungreedy mode (makes quantifiers lazy by default).

Escape Sequences

\\ — Matches a literal backslash.

\t — Matches a tab character.

\n — Matches a newline.

\r — Matches a carriage return.

\f — Matches a form feed.

\v — Matches a vertical tab.

\0 — Matches a null character.

\xhh — Matches the character with the hexadecimal value hh.

\uhhhh — Matches the Unicode character with the hexadecimal value hhhh.

Backreferences

\n — Refers back to the nth capturing group (e.g., \1 refers to the first group).

\k<name> — Refers back to the named capturing group.

Named Groups and References

(?P<name>...) — Creates a named capturing group called name.

(?P=name) — Refers to the previously captured named group.

Assertions

(?=...) — Positive lookahead (asserts that what follows matches the pattern).

(?!...) — Negative lookahead (asserts that what follows does not match the pattern).

(?<=...) — Positive lookbehind (asserts that what precedes matches the pattern).

(?<!...) — Negative lookbehind (asserts that what precedes does not match the pattern).

POSIX Character Classes

[[:digit:]] — Matches digits [0-9].

[[:alpha:]] — Matches letters [a-zA-Z].

[[:alnum:]] — Matches alphanumeric characters [a-zA-Z0-9].

[[:blank:]] — Matches space and tab characters.

[[:space:]] — Matches all whitespace characters.

[[:punct:]] — Matches punctuation characters.

[[:xdigit:]] — Matches hexadecimal digits [0-9a-fA-F].

[[:lower:]] — Matches lowercase letters [a-z].

[[:upper:]] — Matches uppercase letters [A-Z].

Advanced Patterns

(?(condition)yes-pattern|no-pattern) — Conditional expressions. If the condition is true, it matches the yes-pattern; otherwise, it matches the no-pattern.

Unicode Matching

\p{L} — Matches any letter in any language.

\p{N} — Matches any number in any language.

\P{L} — Matches anything that's not a letter.

Common Use Cases

Matching emails: ^\w+@\w+\.\w{2,3}$

Matching phone numbers: \d{3}-\d{3}-\d{4}

Matching URLs: https?:\/\/(www\.)?\w+\.\w+