Argument and Option Handling

The script uses getopts for robust option parsing, handling -n (line numbers) and -v (invert match) flags in any combination (e.g., -nv, -vn). After parsing the options, it shifts past them to access the search string and filename. It validates:

• Presence of the search string and filename (exits with an error if missing).

• File existence (exists if the file is not found).

• The --help flag to display usage information. The search is performed using awk with IGNORECASE=1 for case-insensitive matching. The awk command is built based on the presence of -n and -v flags to include either line numbers or invert the match.

Supporting Regex and Additional Options (-i, -c, -l)

To implement regex, I would adjust the awk pattern to consider the search string as a regular expression (e.g., /$search_string/ instead of matching it exactly). The existing IGNORECASE=1 effectively manages -i (case-insensitive), but it could be toggled with an additional flag. For -c (count matches), I'd incorporate an awk command to increase a counter for matching lines, displaying only the total count. For -l (list filenames), the script must accommodate multiple files and only print the filenames that contain matches. This adjustment would necessitate restructuring the argument parsing to accept several filenames and iterating through them.

Hardest Part to Implement

The hardest part was ensuring flexible option parsing with getopts while maintaining correct argument validation. Handling combinations like -nv and ensuring errors for invalid inputs like if the user missed a search string, required careful logic. Dynamically constructing the awk command to handle all combinations of -n and -v was also challenging, as it needed to be precise to mimic grep's output style.