HyperionDev

Cyber Security Tools – Linux

Model Answer Approach

Visit our website

Auto-graded task

cat

The provided solution emulates the functionality of the `cat` command in Linux, which concatenates and displays the contents of files or reads from standard input until it is closed.

The `std_in()` function reads input from standard input (stdin) until it is closed, mirroring the behaviour of the `cat`command when no file arguments are provided. It iterates over each line of input and prints it to stdout.

The `infile()` function reads from a specified file and prints its contents. It utilises exception handling to catch a FileNotFoundError in case the specified file does not exist, printing an error message accordingly.

In the main block, the script determines whether to read from stdin or a file based on the command-line arguments. If arguments are provided, it calls the infile function with the specified file. Otherwise, it calls the `std_in` function to read from stdin, effectively replicating the behaviour of the `cat` command.

Overall, this solution offers a versatile and concise implementation of the 'cat' command in Python, handling both file input and standard input scenarios.

echo

The provided solution implements a simplified version of the `echo` command in Linux. The `array_to_string()` function converts a list of strings (received as an argument) into a single string. It iterates through each element in the array, concatenating them along with a space separator. The resulting string represents the concatenation of all elements in the input list.

The script removes the name of the program from the list of command-line arguments using the `pop(0)` method, as the echo command does not include the name of the program in its output. Then, it calls the `array_to_string()` function with the modified list of arguments and prints the resulting string to stdout.

This approach mirrors the functionality of the `echo` command by concatenating and printing the provided arguments as a single string. By removing the program name from the list of arguments, the script ensures that only the user-provided arguments are echoed to stdout.

grep

The provided solution aims to implement a simplified version of the `grep` command in Linux. The match function searches for a specified `needle` string within a list of 'source' strings. It iterates through each `haystack` string in the `source` list, utilising the find method to determine if the `needle` string is present within it. If the string is found (indicated by a positive index returned by find), the `haystack` string is printed.

The script accepts command-line arguments to determine whether to read from stdin or from a file. If provided with two arguments, it reads from stdin and executes the match function accordingly. If provided with three arguments, it reads from the specified file and then executes the `match()` function.

This approach ensures flexibility in input sources, allowing users to provide input via stdin or from a file. The match function efficiently searches for the `needle` string within the `source` strings, providing functionality similar to the `grep` command in Linux.