Placement Empowerment Program

Cloud Computing and DevOps Centre

Day 13 – Bulk Search and Replace Tool

Write a script to search and replace words across multiple files using grep and sed.

Name: Hema S

Department: ECE

Introduction

In Linux system administration and scripting, the ability to **automate bulk text operations** across multiple files is highly valuable. The **Bulk Search and Replace Tool** is a simple yet powerful script that utilizes common command-line utilities like **grep, sed**, and **find** to search for specific patterns in a batch of files and replace them automatically. This tool saves time and ensures consistency, especially when managing large codebases or configuration files.

Overview

This Proof of Concept (PoC) demonstrates how to create a Linux shell script that automates the **search and replace operation across multiple files** in a directory. By combining tools like **grep** to identify target files and **sed** to perform replacements, the script efficiently handles batch updates of text content. This is especially useful in tasks like updating configuration values, renaming variables in code, or correcting repeated typos across documents. The script improves accuracy, reduces manual effort, and ensures uniform changes across files

Objectives :

✓ To **automate** the process of searching and replacing text across multiple files.

✓ To **understand and apply** Linux text processing tools like **grep, sed,** and **find.**

✓ To **reduce manual effort** in bulk editing tasks within scripts, configs, or documents.

✓ To build a **reusable shell script** that can be executed with custom inputs.

✓ To enhance Linux scripting skills relevant for **system administration** and **DevOps** tasks.

Importance :

✓ **Saves Time:**
Manually updating text in multiple files is time-consuming. This script automates it in seconds.

✓**Improves Accuracy:**
Reduces the risk of human error during repetitive search-and-replace tasks.

✓**Demonstrates Real-World Utility:**
Such tools are widely used in code refactoring, log sanitization, and config management.

✓**Boosts Scripting Proficiency:**
Enhances your skills with grep, sed, and shell scripting, essential for Linux and DevOps roles.

✓ **Reusable & Scalable:**
Can be reused across projects and scaled to work with large file systems or codebases.

Step-by-Step Overview

Step 1:Open Terminal

Launch a terminal window on your Linux system.

Step 2:Create a Script File

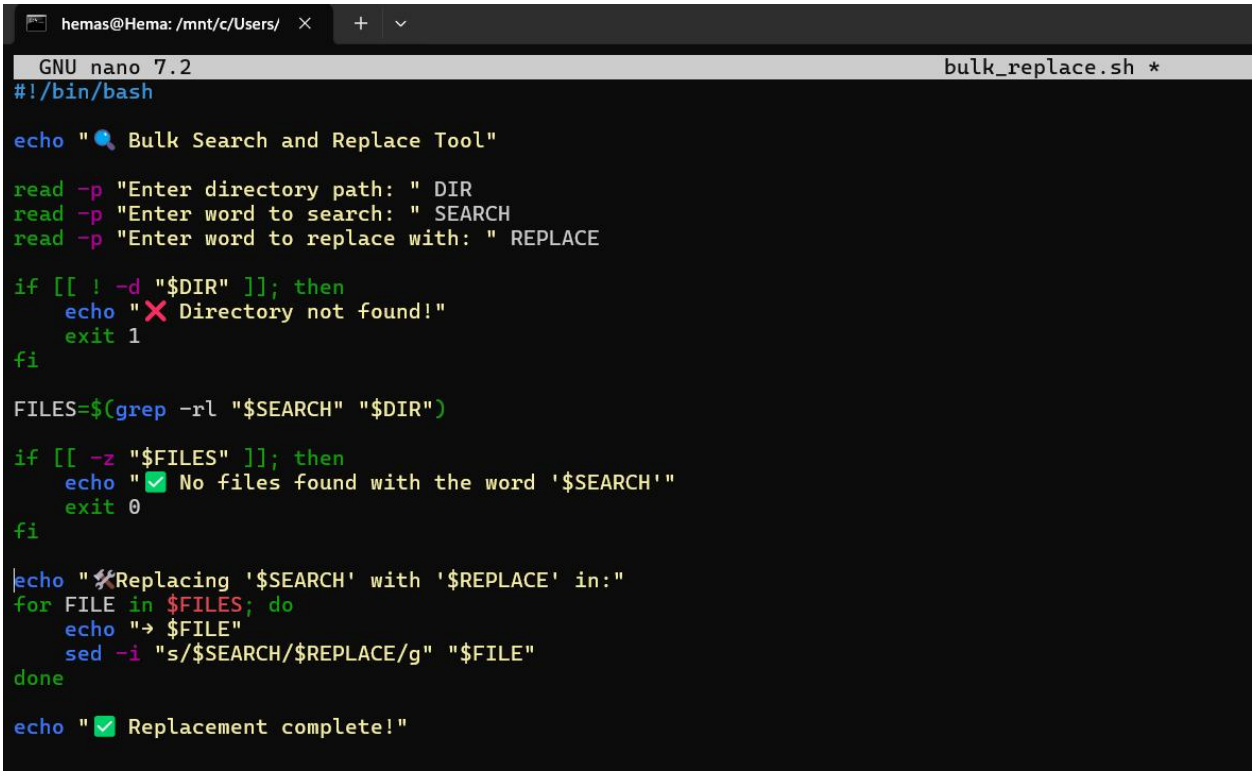
Use nano or any editor to create a shell script:

```
hemas@Hema:/mnt/c/Users/hemas$ nano bulk_replace.sh
```

.

Step 3:Write the Monitoring Script

In the nano editor,Paste the following code:



Step 4:Save and Exit

Press Ctrl + O → Enter (to save)

Press Ctrl + X (to exit)

Step 5: Make the Script Executable

```
hemas@Hema:/mnt/c/Users/hemas$ chmod +x bulk_replace.sh
```

Step 6: Create Sample Files (Optional for Testing)

```
hemas@Hema:/mnt/c/Users/hemas$ mkdir test_folder
hemas@Hema:/mnt/c/Users/hemas$ echo "hello world" > test_folder/file1.txt
hemas@Hema:/mnt/c/Users/hemas$ echo "hello everyone" > test_folder/file2.txt
```

Step 7: Run the Script

```
hemas@Hema:/mnt/c/Users/hemas$ ./bulk_replace.sh
```

Step 8: Provide Inputs When Prompted

Example:

```
🔍 Bulk Search and Replace Tool
Enter directory path: test_folder
Enter word to search: hello
Enter word to replace with: hai
🛠️Replacing 'hello' with 'hai' in:
→ test_folder/file1.txt
→ test_folder/file2.txt
✅ Replacement complete!
```

Step 9: Check the Changes

```
hemas@Hema:/mnt/c/Users/hemas$ cat test_folder/file1.txt
hai world
hemas@Hema:/mnt/c/Users/hemas$ cat test_folder/file2.txt
hai everyone
```

You should see hai world and hai everyone.

Outcomes:

✓ **Created a functional shell script** to automate bulk search and replace across multiple files.

✓ Used **grep -rl** to recursively search for files containing a specific word.

✓ Used **sed -i** to perform in-place replacements of text in matching files.

✓ **Gained practical experience** with:

Shell scripting
Working with loops
Command-line tools (**grep, sed, read, if conditions)**

✓**Understood how to safely modify multiple files** in a directory without doing it manually one by one.

✓**Improved productivity and reduced manual errors**, especially useful for large-scale code refactoring or config updates.

✓ **Built a reusable automation tool** that can be easily modified for other file-processing tasks.