SERVER CONFIG AUTOMATION PROJECT

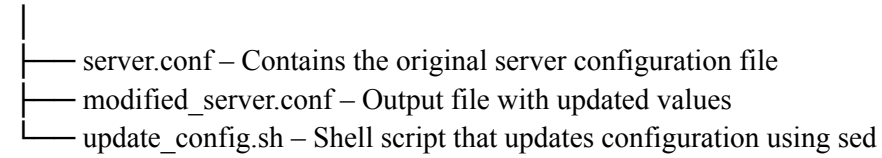
PROBLEM STATEMENT :

The aim of this project is to automate the process of updating server configuration files using the sed command. Since server details like IP addresses, hostnames, and ports change frequently, manual editing is time-consuming and error-prone. This project uses a shell script to automatically replace values, insert comments, append new settings, delete obsolete lines, and preview changes before saving. Automation ensures accuracy, consistency, and efficient server configuration management.

OBJECTIVES

- Automate configuration updates
- Reduce manual errors
- Use sed for editing files
- Preview changes before saving

FOLDER STRUCTURE

OS_PROJECT/


SOLUTION

The solution to the above problem is implemented using a Bash shell script (update_config.sh) and the sed command. The script automates the modification of server configuration files by replacing existing values, inserting new comments, appending missing settings, deleting outdated entries, and previewing changes before applying them.

➔ Step 1 : Creating the Server Configuration File

This file contains server configuration parameters such as server_ip, hostname, port, max_threads, log_level, max_connections, and timeout.

```
# Configuration updated on Mon Dec 15 10:50:17 IST 2025
SERVER_IP=10.0.0.50
HOSTNAME=production-db
PORT=5432
MAX_THREADS=50
LOG_LEVEL=info
MAX_CONNECTIONS=100
TIMEOUT=30
```

➔ Step 2 : Create Shell Script

Shell script named update_config.sh is created to automate configuration updates.
The script performs the following operations

```
#!/bin/bash

INPUT_FILE="server.conf"
OUTPUT_FILE="modified_server.conf"

echo "Server Configuration Update Script"
echo "----------------------------------"

read -p "Enter NEW Server IP: " NEW_IP
read -p "Enter NEW Hostname: " NEW_HOST
read -p "Enter NEW Port: " NEW_PORT
c
read -p "Enter MAX_CONNECTIONS value: " MAX_CONN
read -p "Enter TIMEOUT value: " TIMEOUT_VAL

echo ""
echo " Previewing changes"
echo "-----------------------------------"

sed -n "s/^SERVER_IP=.*/SERVER_IP=$NEW_IP/p" "$INPUT_FILE"
sed -n "s/^HOSTNAME=.*/HOSTNAME=$NEW_HOST/p" "$INPUT_FILE"
sed -n "s/^PORT=.*/PORT=$NEW_PORT/p" "$INPUT_FILE"

echo ""
read -p "Apply these changes? (y/n): " CONFIRM
[ "$CONFIRM" != "y" ] && echo " Cancelled" && exit 0

echo " Generating modified_server.conf ..."

sed \
  -e '/^# Configuration updated/d' \
  -e "s/^SERVER_IP=.*/SERVER_IP=$NEW_IP/" \
  -e "s/^HOSTNAME=.*/HOSTNAME=$NEW_HOST/" \
  -e "s/^PORT=.*/PORT=$NEW_PORT/" \
  -e "s/^MAX_CONNECTIONS=.*/MAX_CONNECTIONS=$MAX_CONN/" \
  -e "s/^TIMEOUT=.*/TIMEOUT=$TIMEOUT_VAL/" \
  "$INPUT_FILE" > "$OUTPUT_FILE"

# Append only if missing
grep -q '^MAX_CONNECTIONS=' "$OUTPUT_FILE" || echo "MAX_CONNECTIONS=$MAX_CONN" >> "$OUTPUT_FILE"
grep -q '^TIMEOUT=' "$OUTPUT_FILE" || echo "TIMEOUT=$TIMEOUT_VAL" >> "$OUTPUT_FILE"

# Insert fresh comment
sed -i '' "1i\\
# Configuration updated on $(date)
" "$OUTPUT_FILE"

echo ""
echo " Final modified_server.conf:"
echo "-------------------------------"
sed -n 'p' "$OUTPUT_FILE"

echo " Done!"
```

➔ Step 3 : Reading User Input

The script uses the `read` command to accept new configuration values from the user and dynamic updates without manually editing the configuration file.

```
 9 read -p "Enter NEW Server IP: " NEW_IP
10 read -p "Enter NEW Hostname: " NEW_HOST
11 read -p "Enter NEW Port: " NEW_PORT
12 c
13 read -p "Enter MAX_CONNECTIONS value: " MAX_CONN
14 read -p "Enter TIMEOUT value: " TIMEOUT_VAL
15
```

➔ Step 4 : Previewing Changes Using sed -n and p

Before applying any changes, the script previews the updated values using the **`sed` substitution (`s`)** command along with the **print (`p`)** option.

```
20 sed -n "s/^SERVER_IP=.*/SERVER_IP=$NEW_IP/p" "$INPUT_FILE"
21 sed -n "s/^HOSTNAME=.*/HOSTNAME=$NEW_HOST/p" "$INPUT_FILE"
22 sed -n "s/^PORT=.*/PORT=$NEW_PORT/p" "$INPUT_FILE"
23
24 echo ""
25 read -p "Apply these changes? (y/n): " CONFIRM
26 [ "$CONFIRM" != "y" ] && echo " Cancelled" && exit 0
```

➔ Step 5 : Applying Changes Using sed

After user confirmation, the script uses multiple sed commands like s for substitution, i for insertion, a for appending values, d for deletion, and p for previewing changes.

```
30 sed \
31   -e '/^# Configuration updated/d' \
32   -e "s/^SERVER_IP=.*/SERVER_IP=$NEW_IP/" \
33   -e "s/^HOSTNAME=.*/HOSTNAME=$NEW_HOST/" \
34   -e "s/^PORT=.*/PORT=$NEW_PORT/" \
35   -e "s/^MAX_CONNECTIONS=.*/MAX_CONNECTIONS=$MAX_CONN/" \
36   -e "s/^TIMEOUT=.*/TIMEOUT=$TIMEOUT_VAL/" \
37   "$INPUT_FILE" > "$OUTPUT_FILE"
38
39 # Append only if missing
40 grep -q '^MAX_CONNECTIONS=' "$OUTPUT_FILE" || echo "MAX_CONNECTIONS=$MAX_CONN" >> "$OUTPUT_FILE"
41 grep -q '^TIMEOUT=' "$OUTPUT_FILE" || echo "TIMEOUT=$TIMEOUT_VAL" >> "$OUTPUT_FILE"
42
```

➔ Step 6 : Inserting Comment and Displaying Final Output

After applying the configuration changes, the script uses the sed insert (i) command to add a fresh timestamp comment at the beginning of the output file

```
# Insert fresh comment
sed -i '' "1i\\
# Configuration updated on $(date)
" "$OUTPUT_FILE"

echo ""
echo " Final modified_server.conf:"
echo "------------------------------"
sed -n 'p' "$OUTPUT_FILE"

echo " Done!"
```

➔ Step 7 : Executing the Shell Script

```
[mac@MACs-MacBook-Air Os_Project % chmod +x update_config.sh
[mac@MACs-MacBook-Air Os_Project % bash update_config.sh
 Server Configuration Update Script
-----------------------------------
Enter NEW Server IP: 20.0.0.10
Enter NEW Hostname: amazon.com
Enter NEW Port: 20
Enter MAX_CONNECTIONS value: 12
Enter TIMEOUT value: 20

 Previewing changes
-----------------------------------
SERVER_IP=20.0.0.10
HOSTNAME=amazon.com
PORT=20

Apply these changes? (y/n): y
Generating modified_server.conf ...

 Final modified_server.conf:
-------------------------------
# Configuration updated on Tue Dec 16 12:44:55 IST 2025
SERVER_IP=20.0.0.10
HOSTNAME=amazon.com
PORT=20
MAX_THREADS=50
LOG_LEVEL=info
MAX_CONNECTIONS=12
TIMEOUT=20
 Done!
mac@MACs-MacBook-Air Os_Project %
```

FINAL RESULT

The server configuration file is successfully updated using automated `sed` commands.

The solution eliminates manual editing, reduces errors, and ensures consistent configuration management.

Repository link : https://github.com/avi-0605/Server_config_automation_OS