Experiment 11: Shell Programming – Advanced String & File Operations

Name: Aditya Mishra

Roll No.: 590029219

Date: 2025-11-05

Aim

To write and execute shell scripts for advanced string manipulation, file processing, menu-driven systems, and dictionary-based word validation.

Requirements

- Linux system with Bash shell
- Commands: `grep`, `cut`, `awk`, `rev`, `tr`, `df`, `free`, `cal`, `date`
- Dictionary file at `/usr/share/dict/words` (for dictionary check task)

Exercise 1: Split Sentence into Words

Task Statement

Write a shell script to split a sentence into individual words and print them one per line.

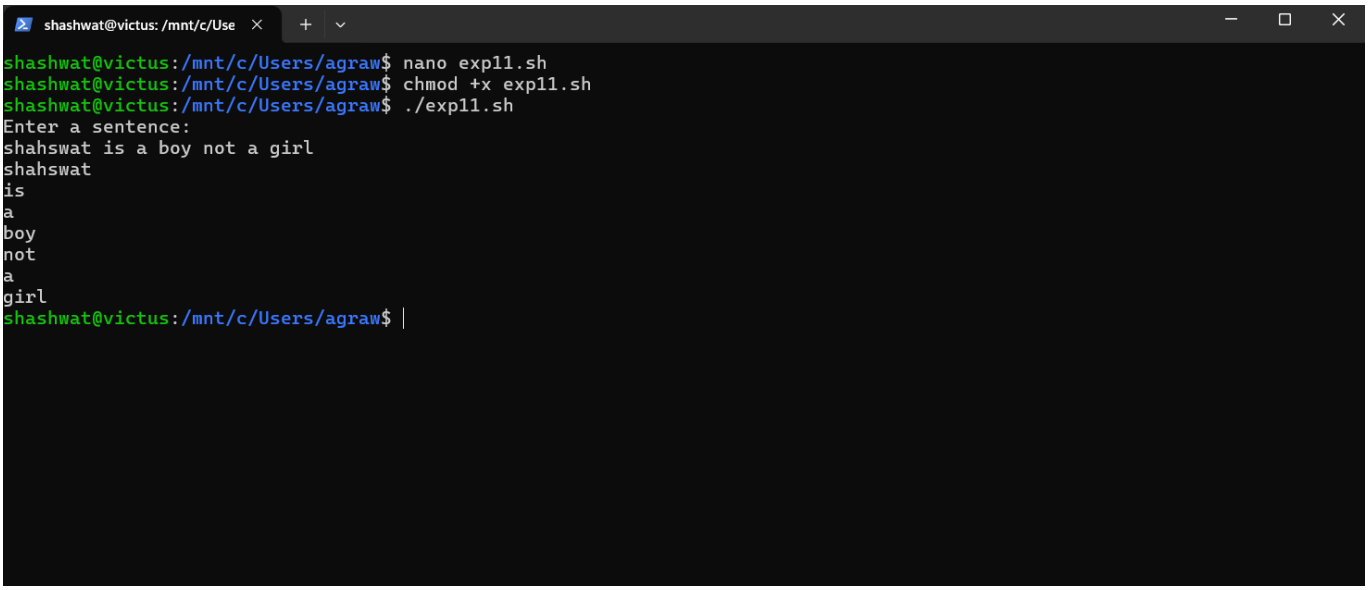
Script

```
#!/bin/bash
echo "Enter a sentence:"
read sentence
for word in $sentence; do
echo "$word"
done
```

Explanation

- When `$sentence` is **unquoted**, Bash performs word splitting automatically based on the **IFS (Internal Field Separator)**.
- By default, it splits on spaces, tabs, and newlines.
- Each word becomes a separate iteration of the loop.

Output



Exercise 2: Palindrome Check

Task Statement

Write a script that checks whether a given string is a palindrome.

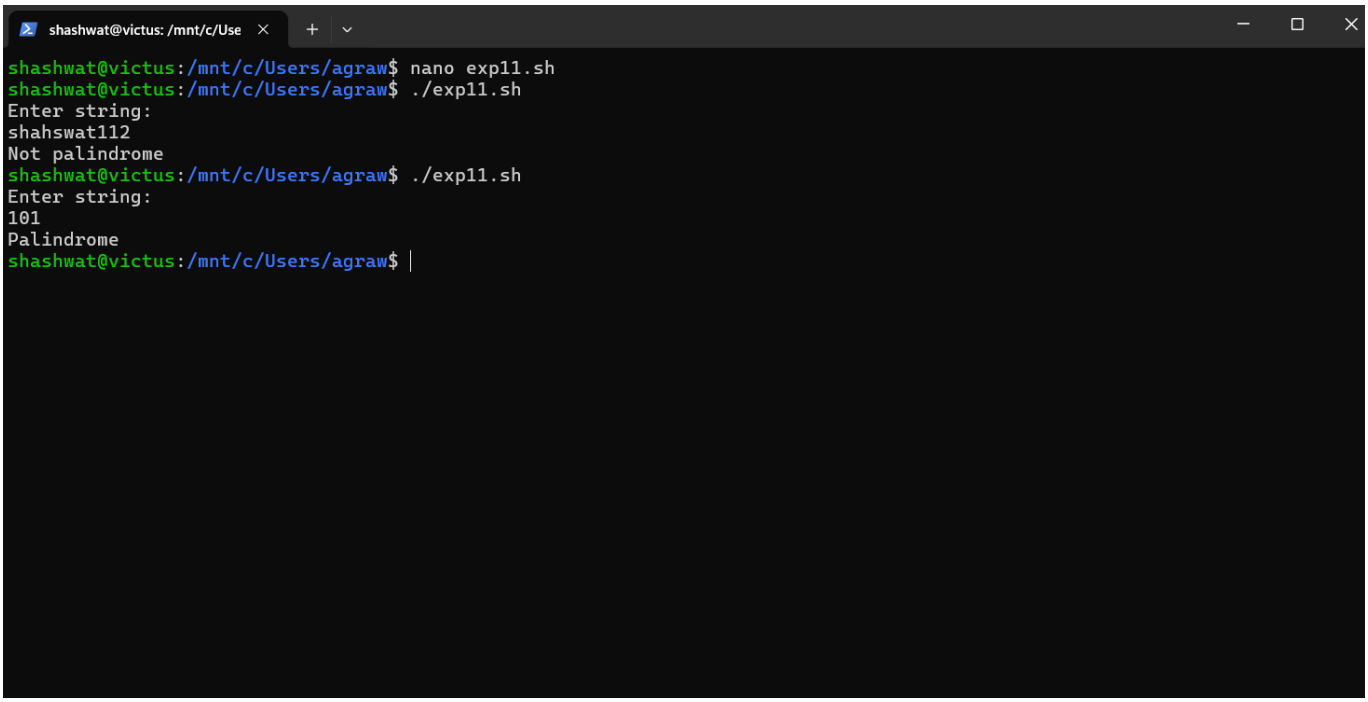
Script

```
#!/bin/bash
echo "Enter string:"
read str
rev=$(echo "$str" | rev)
if [ "$str" = "$rev" ]; then
echo "Palindrome"
else
echo "Not palindrome"
fi
```

Explanation

- The `rev` command reverses a string character by character.
- Comparison checks if the original string equals the reversed one.
- This comparison is **case-sensitive** and includes spaces.

Output



Exercise 3: CSV File Processing – Print First Column

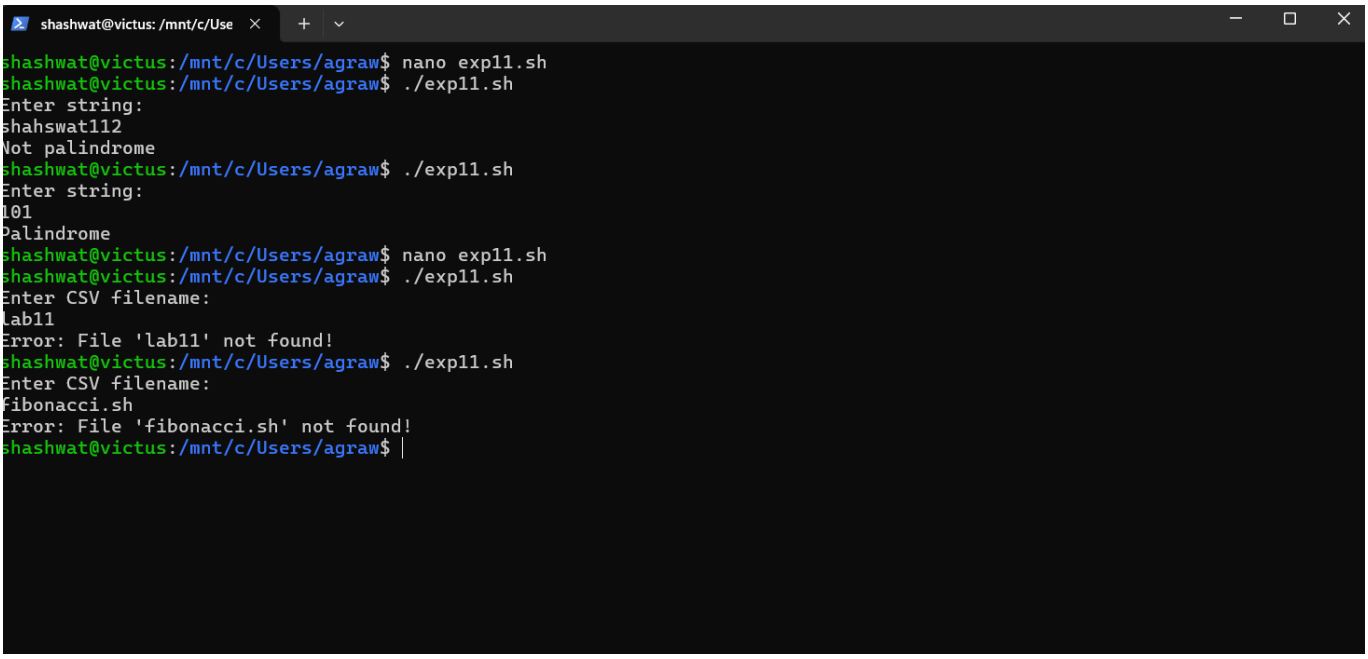
Task Statement

Write a script to print the first column of a CSV file provided by the user.

Script

```
#!/bin/bash
echo "Enter CSV filename:"
read filename
if [ ! -f "$filename" ]; then
echo "Error: File '$filename' not found!"
exit 1
fi
echo "First column values:"
echo "--------------------"
cut -d',' -f1 "$filename"
```

Output



Exercise 4: Interactive Menu System

Task Statement

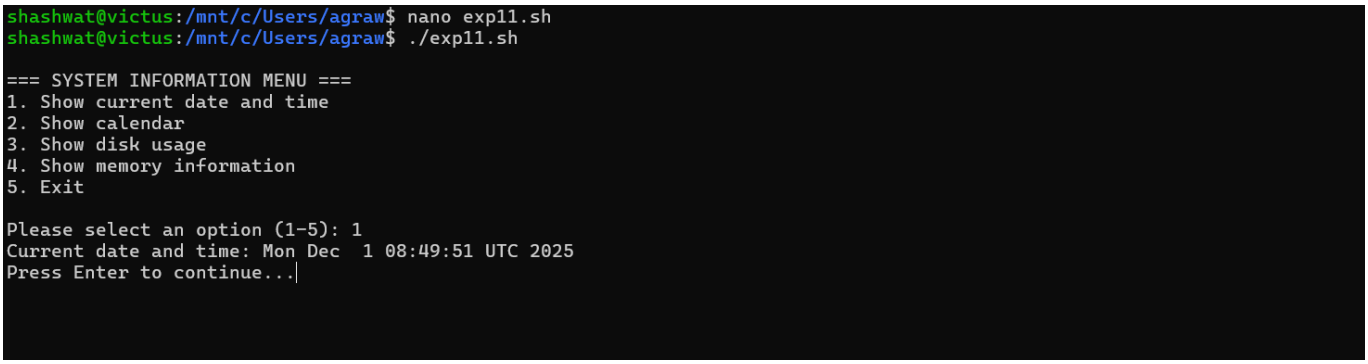
Create a menu-driven script that allows users to display system information.

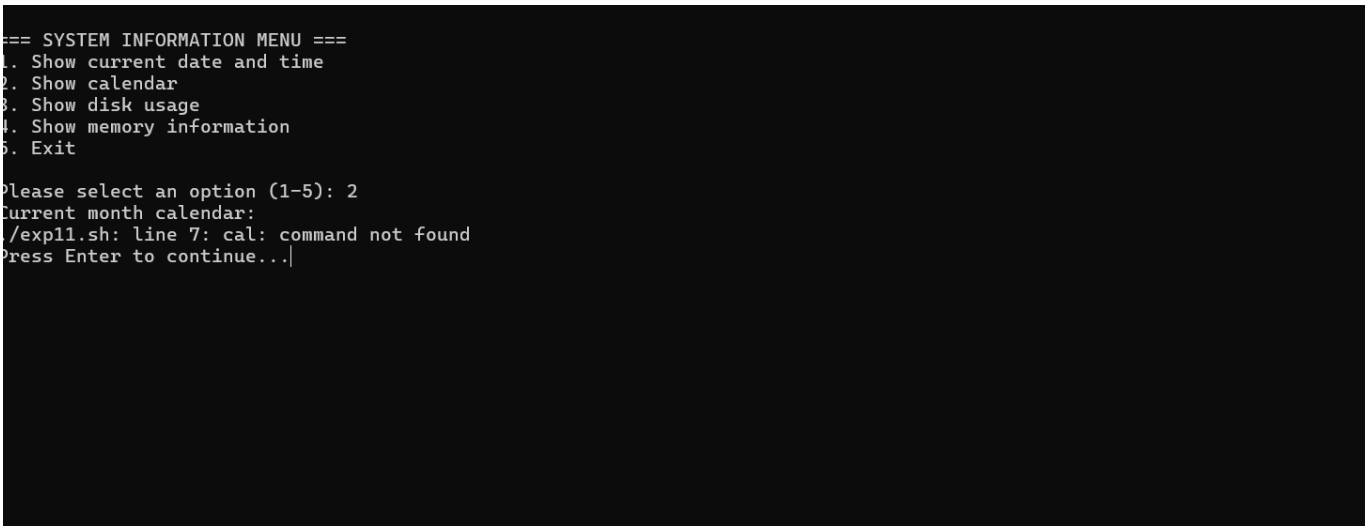
Script

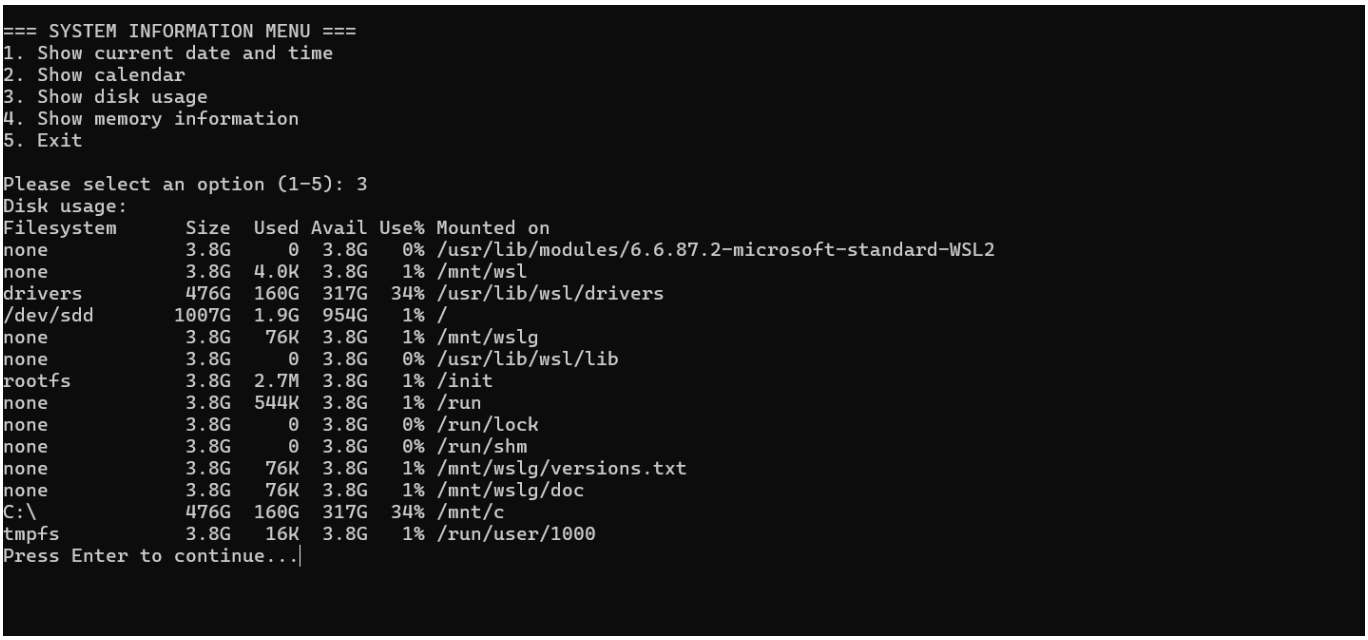
```
#!/bin/bash
show_date() {
echo "Current date and time: $(date)"
}
show_calendar() {
echo "Current month calendar:"
cal
}
show_disk_usage() {
echo "Disk usage:"
df -h
}
show_memory_info() {
echo "Memory information:"
free -h
}
while true; do
echo ""
echo "=== SYSTEM INFORMATION MENU ==="
echo "1. Show current date and time"
echo "2. Show calendar"
echo "3. Show disk usage"
echo "4. Show memory information"
echo "5. Exit"
```

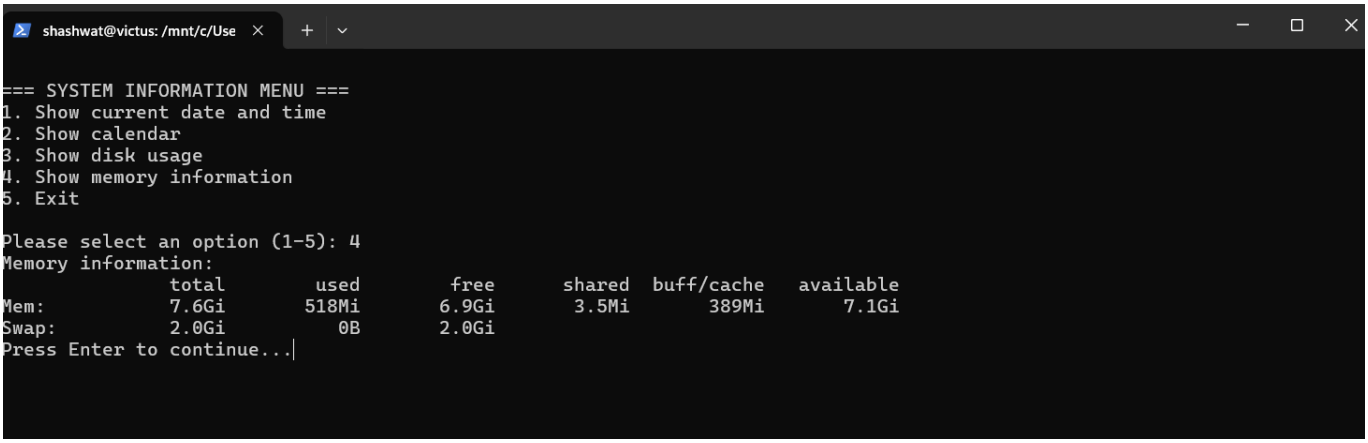
```
  echo ""
  read -p "Please select an option (1-5): " choice
  case $choice in
  1) show_date ;;
  2) show_calendar ;;
  3) show_disk_usage ;;
  4) show_memory_info ;;
  5) echo "Goodbye!"; break ;;
  *) echo "Invalid option! Please enter a number between 1-5." ;;
  esac
  read -p "Press Enter to continue..."
  clear
  done
```

Output









Exercise 5: Dictionary Word Check

Task Statement

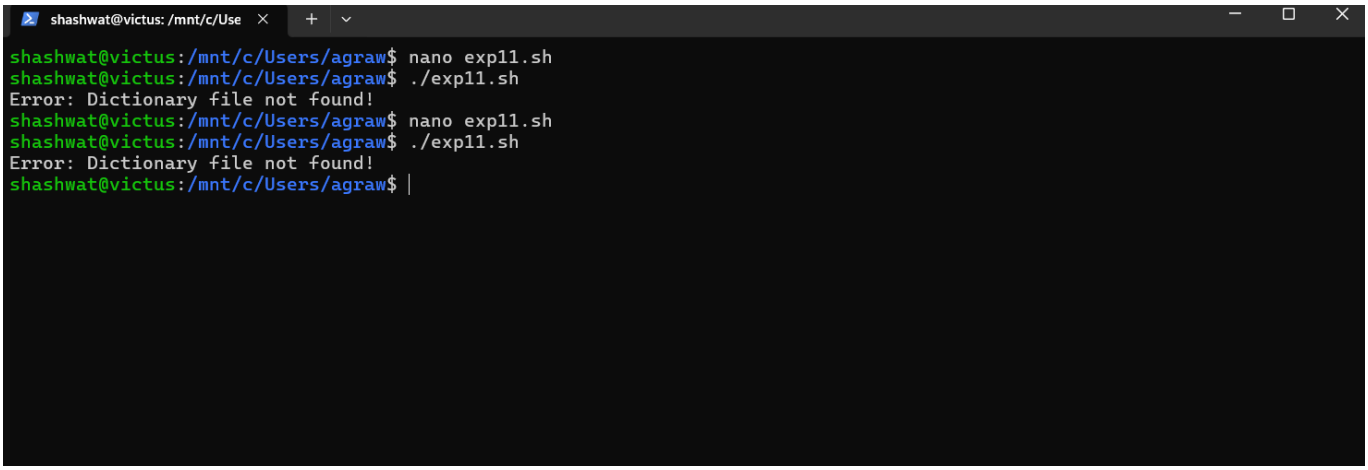
Write a script to check if a word exists in the system dictionary.

Script

```
#!/bin/bash
DICTIONARY="/usr/share/dict/words"
if [ ! -f "$DICTIONARY" ]; then
echo "Error: Dictionary file not found!"
```

```
exit 1
fi
echo "Enter a word to check:"
read word
word_lower=$(echo "$word" | tr '[:upper:]' '[:lower:]')
if grep -q "^${word_lower}$" "$DICTIONARY"; then
echo "3 '$word' is a valid English word."
else
echo "7 '$word' is not found in the dictionary."
echo ""
echo "Similar words:"
grep -i "^${word:0:3}" "$DICTIONARY" | head -5
fi
```

output:



Explanation

- `grep -q` performs a quiet search for an exact match.
- Suggestions are generated by finding words starting with the first 3 letters.
- Useful for spell checking and learning shell-based text processing.

Result

Successfully implemented shell scripts for sentence splitting, palindrome checking, CSV processing, interactive menus, and dictionary-based word verification.

Conclusion

This experiment strengthened understanding of **string handling, user interaction, text processing, and file manipulation** in Bash scripting.