

Experiment 8: Shell Programming (Continued)

Name: Tanishq bansal Roll No.: 590029349 Date: 2025-10-29

Aim:

- To extend shell programming concepts by using conditional statements, advanced scripting constructs, and command-line arguments.
- To practice writing scripts that perform decision-making and parameter handling.

Requirements

- A Linux system with bash shell.
- Text editor and permission to create/execute shell scripts.

Theory

Conditional execution in shell scripts allows branching logic using `if`, `elif`, `else`, and `case` statements. Scripts can accept command-line arguments using `$1`, `$2`, ... and `$@` for all arguments. Control flow constructs combined with user input and arguments allow dynamic and reusable scripts.

Procedure & Observations

Exercise 1: Using `if-else`

Task Statement:

Write a script to check whether a given number is positive, negative, or zero.

Explanation:

We used an `if-elif-else` construct to compare the number against 0.

Command(s):

```
#!/bin/bash
num=$1
if [ $num -gt 0 ]; then
    echo "$num is positive"
elif [ $num -lt 0 ]; then
    echo "$num is negative"
else
    echo "$num is zero"
fi
```

Output:

```

tanishq@Tanishq: /mnt/c/Use x + v
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/LINUX PRO/day 8$ vim bro.sh
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/LINUX PRO/day 8$ bash bro.sh
bro.sh: line 3: [: -gt: unary operator expected
bro.sh: line 5: [: -lt: unary operator expected
is zero
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/LINUX PRO/day 8$ cat bro.sh
#!/bin/bash
num=$1
if [ $num -gt 0 ]; then
    echo "$num is positive"
elif [ $num -lt 0 ]; then
    echo "$num is negative"
else
    echo "$num is zero"
fi
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/LINUX PRO/day 8$ |

```

Exercise 2: Using `case`

Task Statement:

Write a script that takes a character as input and classifies it as vowel, consonant, digit, or special character.

Explanation:

The `case` statement provides pattern matching for multiple options.

Command(s):

```

#!/bin/bash
ch=$1
case $ch in
    [aeiouAEIOU]) echo "$ch is a vowel" ;;
    [bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ]) echo "$ch is a consonant" ;;
    [0-9]) echo "$ch is a digit" ;;
    *) echo "$ch is a special character" ;;
esac

```

Output:

```

tanishq@Tanishq: /mnt/c/Use x + v
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/LINUX PRO/day 8$ vim bro1.sh
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/LINUX PRO/day 8$ bash bro1.sh
is a special character
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/LINUX PRO/day 8$ |

```

Exercise 3: Command-line arguments

Task Statement:

Write a script that accepts filename(s) as arguments and prints the number of lines in each file.

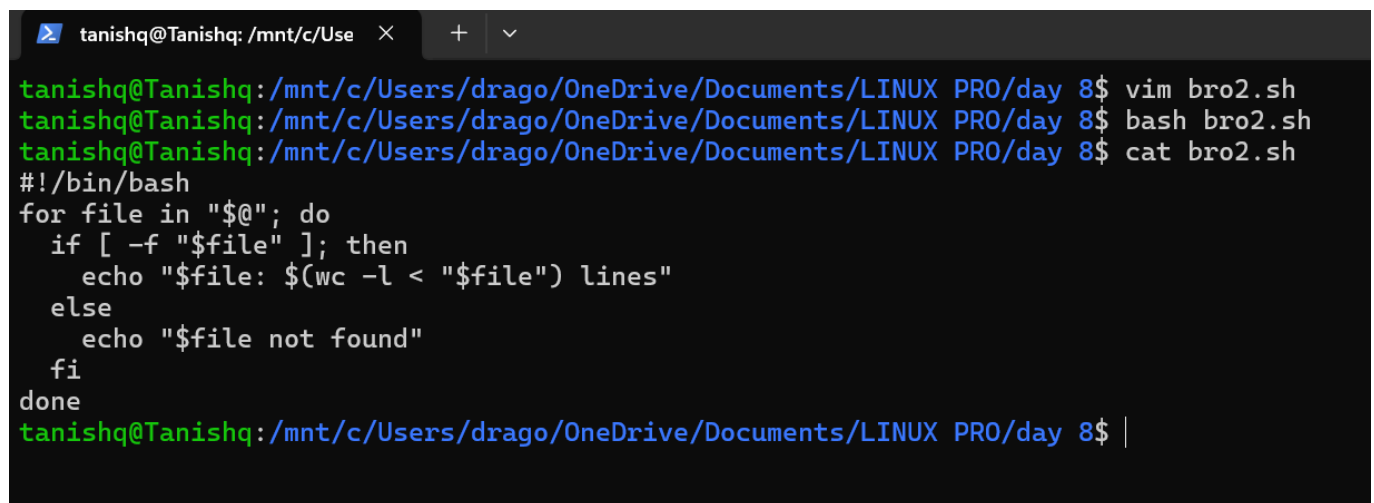
Explanation:

Command-line arguments are accessed using `$@`. Looping through each argument allows file-wise operations.

Command(s):

```
#!/bin/bash
for file in "$@"; do
    if [ -f "$file" ]; then
        echo "$file: $(wc -l < "$file") lines"
    else
        echo "$file not found"
    fi
done
```

Output:

A terminal window with a dark background. The prompt is 'tanishq@Tanishq: /mnt/c/Use'. The user enters 'vim bro2.sh', then 'bash bro2.sh', and finally 'cat bro2.sh'. The output of the script is displayed, showing the number of lines for each file. The script content is also visible in the terminal output.

```
tanishq@Tanishq: /mnt/c/Users/drago/OneDrive/Documents/LINUX PRO/day 8$ vim bro2.sh
tanishq@Tanishq: /mnt/c/Users/drago/OneDrive/Documents/LINUX PRO/day 8$ bash bro2.sh
tanishq@Tanishq: /mnt/c/Users/drago/OneDrive/Documents/LINUX PRO/day 8$ cat bro2.sh
#!/bin/bash
for file in "$@"; do
    if [ -f "$file" ]; then
        echo "$file: $(wc -l < "$file") lines"
    else
        echo "$file not found"
    fi
done
tanishq@Tanishq: /mnt/c/Users/drago/OneDrive/Documents/LINUX PRO/day 8$ |
```

Exercise 4: Nested conditionals

Task Statement:

Write a script to check if a year is a leap year.

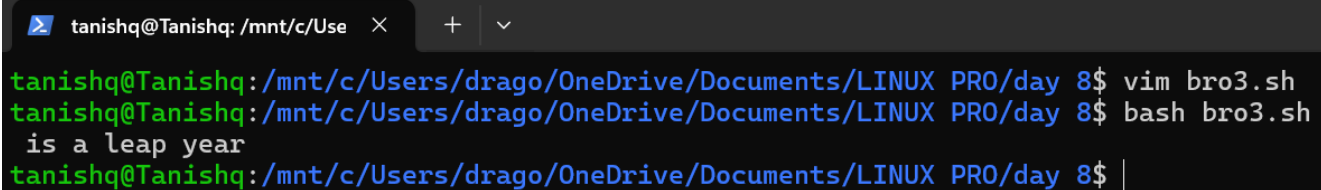
Explanation:

A leap year is divisible by 4, but if divisible by 100 it must also be divisible by 400.

Command(s):

```
#!/bin/bash
year=$1
if (( year % 400 == 0 )); then
    echo "$year is a leap year"
elif (( year % 100 == 0 )); then
    echo "$year is not a leap year"
elif (( year % 4 == 0 )); then
    echo "$year is a leap year"
else
    echo "$year is not a leap year"
fi
```

Output:

A terminal window with a dark background. The prompt is 'tanishq@Tanishq: /mnt/c/Use'. The user enters 'vim bro3.sh' and then 'bash bro3.sh'. The output of the script is 'is a leap year'.

```
tanishq@Tanishq: /mnt/c/Use × + ▾
tanishq@Tanishq: /mnt/c/Users/drago/OneDrive/Documents/LINUX PRO/day 8$ vim bro3.sh
tanishq@Tanishq: /mnt/c/Users/drago/OneDrive/Documents/LINUX PRO/day 8$ bash bro3.sh
is a leap year
tanishq@Tanishq: /mnt/c/Users/drago/OneDrive/Documents/LINUX PRO/day 8$ |
```

Result

- Implemented conditional statements (`if-else`, `case`) in shell scripts.
- Practiced handling command-line arguments and nested conditions.
- Wrote reusable and flexible shell scripts.

Challenges Faced & Learning Outcomes

- Challenge 1: Forgetting to quote variables in conditions — resolved by using `"$var"` to avoid word splitting.
- Challenge 2: Pattern matching in `case` — practiced with multiple examples.

Learning:

- Learned practical use of branching and decision-making in shell scripting.
- Understood command-line argument handling for automation.

Conclusion

This experiment extended shell programming by introducing decision-making and parameter handling. The scripts demonstrate the flexibility of shell programming for different use cases.