

Experiment 6: Shell Loops

Name:Aarush Prasad **Roll No.:** 590027630 **Date:** 2025-10-30

Aim:

- To understand and implement shell loops (`for` , `while` , `until`) in Bash.
- To practice loop control constructs (`break` , `continue`) and loop-based file processing.

Requirements

- A Linux system with bash shell.
- A text editor (nano, vim) and permission to create and execute shell scripts.

Theory

Loops allow repeated execution of commands until a condition is met. Common loop constructs in Bash include `for` (iterate over items), `while` (repeat while condition true), and `until` (repeat until condition becomes true). Loop control statements like `break` and `continue` change the flow inside loops. Loops are essential for automating repetitive tasks such as processing multiple files, generating sequences, and collecting user input.

Procedure & Observations

Exercise 1: Simple `for` loop

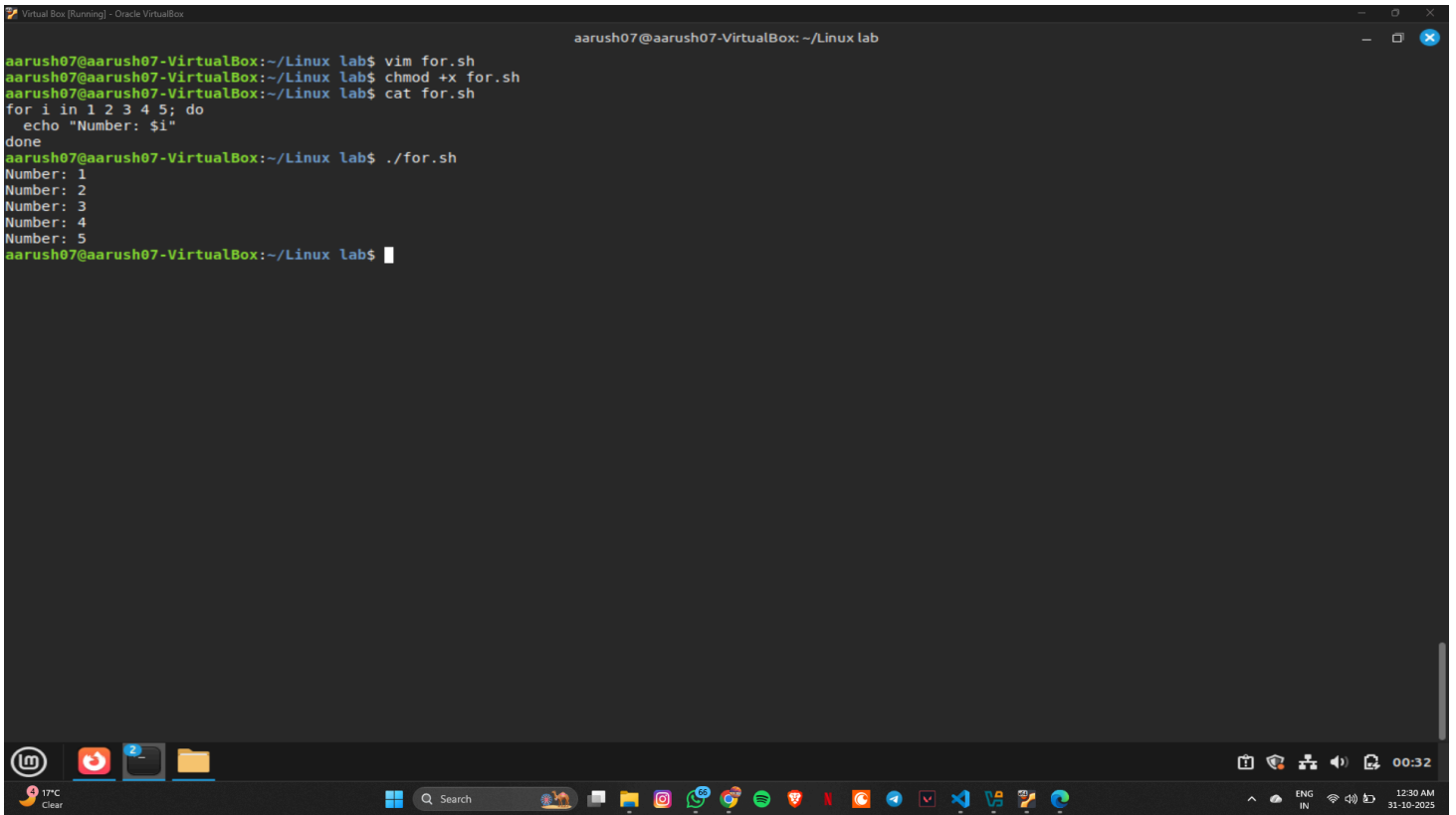
Task Statement:

Write a `for` loop that prints numbers 1 to 5.

Command(s):

```
for i in 1 2 3 4 5; do
    echo "Number: $i"
done
```

Output:



The screenshot shows a terminal window titled "Virtual Box [Running] - Oracle VM VirtualBox". The user is logged in as "aarush07" and is in the directory "~/Linux lab". The terminal shows the following commands and output:

```
aarush07@aarush07-VirtualBox:~/Linux lab$ vim for.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ chmod +x for.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ cat for.sh
for i in 1 2 3 4 5; do
    echo "Number: $i"
done
aarush07@aarush07-VirtualBox:~/Linux lab$ ./for.sh
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
aarush07@aarush07-VirtualBox:~/Linux lab$
```

The terminal window has a dark background with green and blue text. The bottom of the window shows a taskbar with various application icons and system status information, including the date and time (12:30 AM, 31-10-2025).

Exercise 2: for loop over files

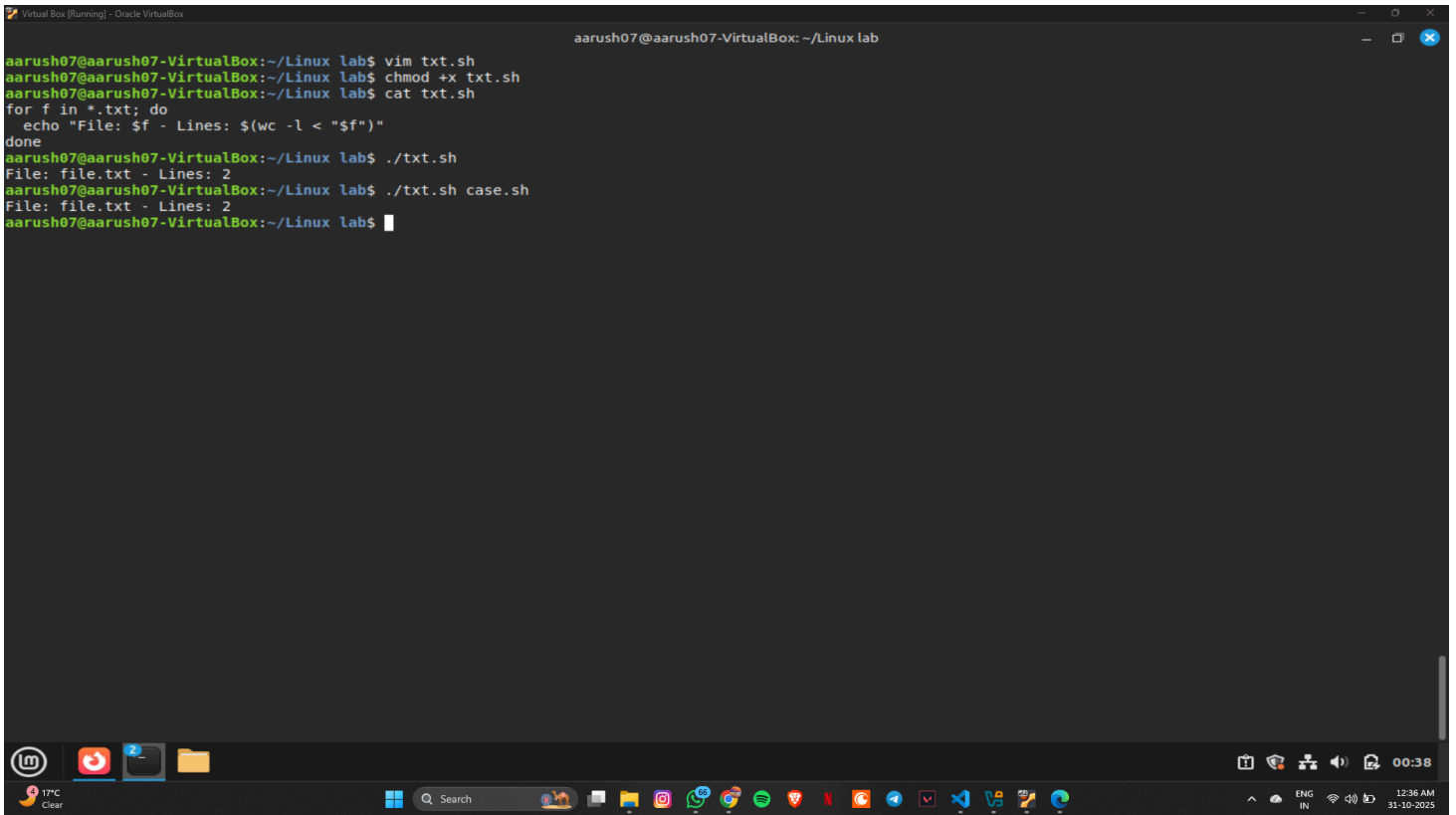
Task Statement:

Process all `.txt` files in a directory and count lines in each.

Command(s):

```
for f in *.txt; do
    echo "File: $f - Lines: $(wc -l < "$f")"
done
```

Output:



```
aarush07@aarush07-VirtualBox: ~/Linux lab$ vim txt.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ chmod +x txt.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ cat txt.sh
for f in *.txt; do
    echo "File: $f - Lines: $(wc -l < "$f")"
done
aarush07@aarush07-VirtualBox:~/Linux lab$ ./txt.sh
File: file.txt - Lines: 2
aarush07@aarush07-VirtualBox:~/Linux lab$ ./txt.sh case.sh
File: file.txt - Lines: 2
aarush07@aarush07-VirtualBox:~/Linux lab$
```

Exercise 3: C-style for loop

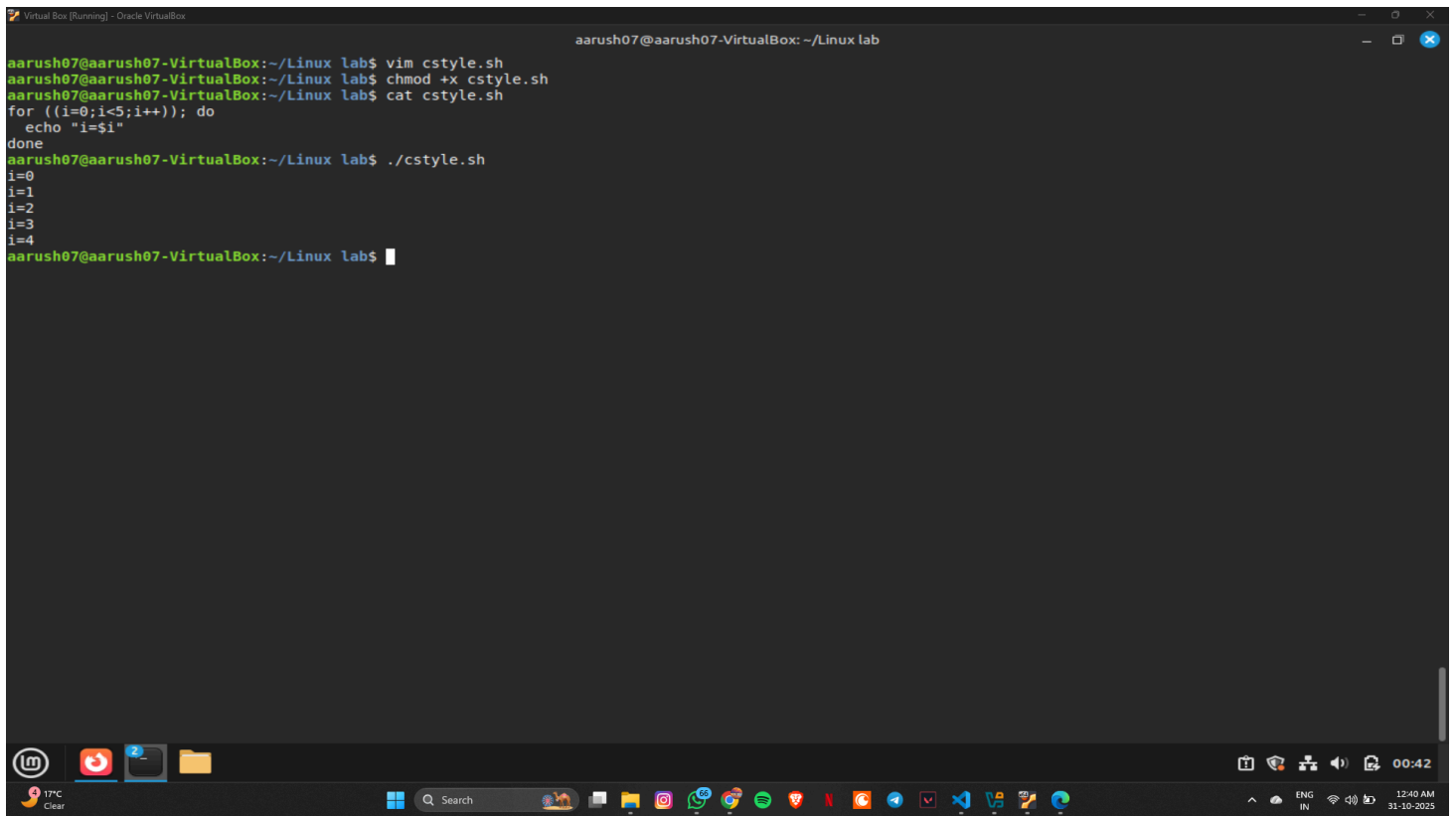
Task Statement:

Use arithmetic C-style loop for numeric iteration.

Command(s):

```
for ((i=0;i<5;i++)); do
    echo "i=$i"
done
```

Output:



The screenshot shows a terminal window titled "Virtual Box [Running] - Oracle VM VirtualBox" with the user "aarush07" at the prompt "aarush07@aarush07-VirtualBox: ~/Linux lab". The terminal displays the following commands and output:

```
aarush07@aarush07-VirtualBox:~/Linux lab$ vim cstyle.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ chmod +x cstyle.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ cat cstyle.sh
for ((i=0;i<5;i++)); do
    echo "i=$i"
done
aarush07@aarush07-VirtualBox:~/Linux lab$ ./cstyle.sh
i=0
i=1
i=2
i=3
i=4
aarush07@aarush07-VirtualBox:~/Linux lab$
```

The bottom of the image shows a Windows taskbar with various application icons, a search bar, and system tray icons including the date and time (12:40 AM, 31-10-2025).

Exercise 4: while loop and reading input

Task Statement:

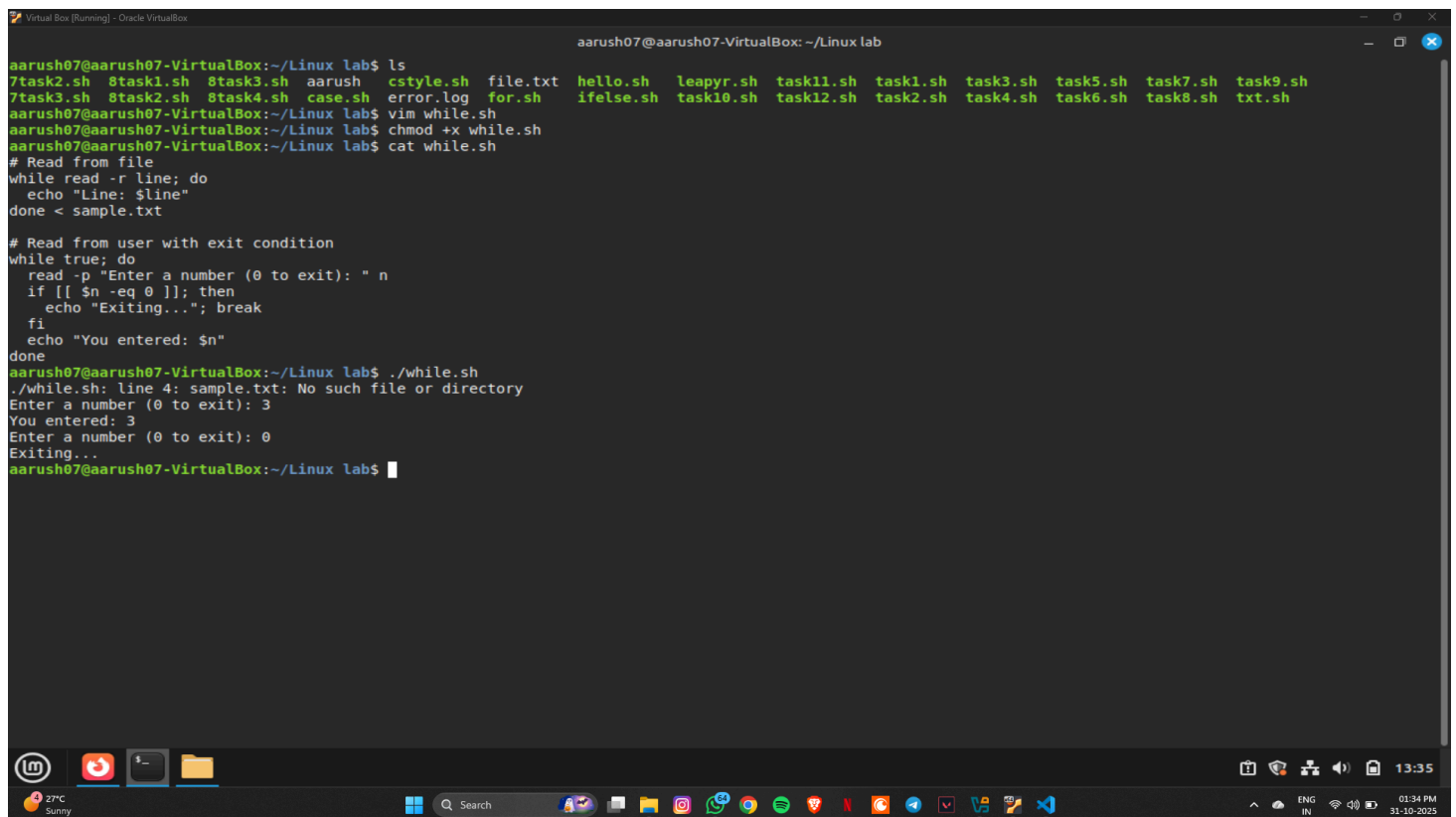
Write a `while` loop that reads lines from a file or from user input.

Command(s):

```
# Read from file
while read -r line; do
    echo "Line: $line"
done < sample.txt

# Read from user with exit condition
while true; do
    read -p "Enter a number (0 to exit): " n
    if [[ $n -eq 0 ]]; then
        echo "Exiting..."; break
    fi
    echo "You entered: $n"
done
```

Output:



The screenshot shows a terminal window titled "Virtual Box [Running] - Oracle VM VirtualBox" with the prompt "aarush07@aarush07-VirtualBox: ~/Linux lab". The terminal displays the following commands and output:

```
aarush07@aarush07-VirtualBox:~/Linux lab$ ls
7task2.sh 8task1.sh 8task3.sh aarush  cstyle.sh file.txt hello.sh leapyr.sh task11.sh task1.sh task3.sh task5.sh task7.sh task9.sh
7task3.sh 8task2.sh 8task4.sh case.sh error.log for.sh ifelse.sh task10.sh task12.sh task2.sh task4.sh task6.sh task8.sh txt.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ vim while.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ chmod +x while.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ cat while.sh
# Read from file
while read -r line; do
    echo "Line: $line"
done < sample.txt

# Read from user with exit condition
while true; do
    read -p "Enter a number (0 to exit): " n
    if [[ $n -eq 0 ]]; then
        echo "Exiting..."; break
    fi
    echo "You entered: $n"
done
aarush07@aarush07-VirtualBox:~/Linux lab$ ./while.sh
./while.sh: line 4: sample.txt: No such file or directory
Enter a number (0 to exit): 3
You entered: 3
Enter a number (0 to exit): 0
Exiting...
aarush07@aarush07-VirtualBox:~/Linux lab$
```

The terminal window is part of a desktop environment. The taskbar at the bottom shows various application icons, including a web browser, file explorer, and communication tools. The system tray on the right indicates the date and time as 01:34 PM on 31-10-2025.

Exercise 5: until loop

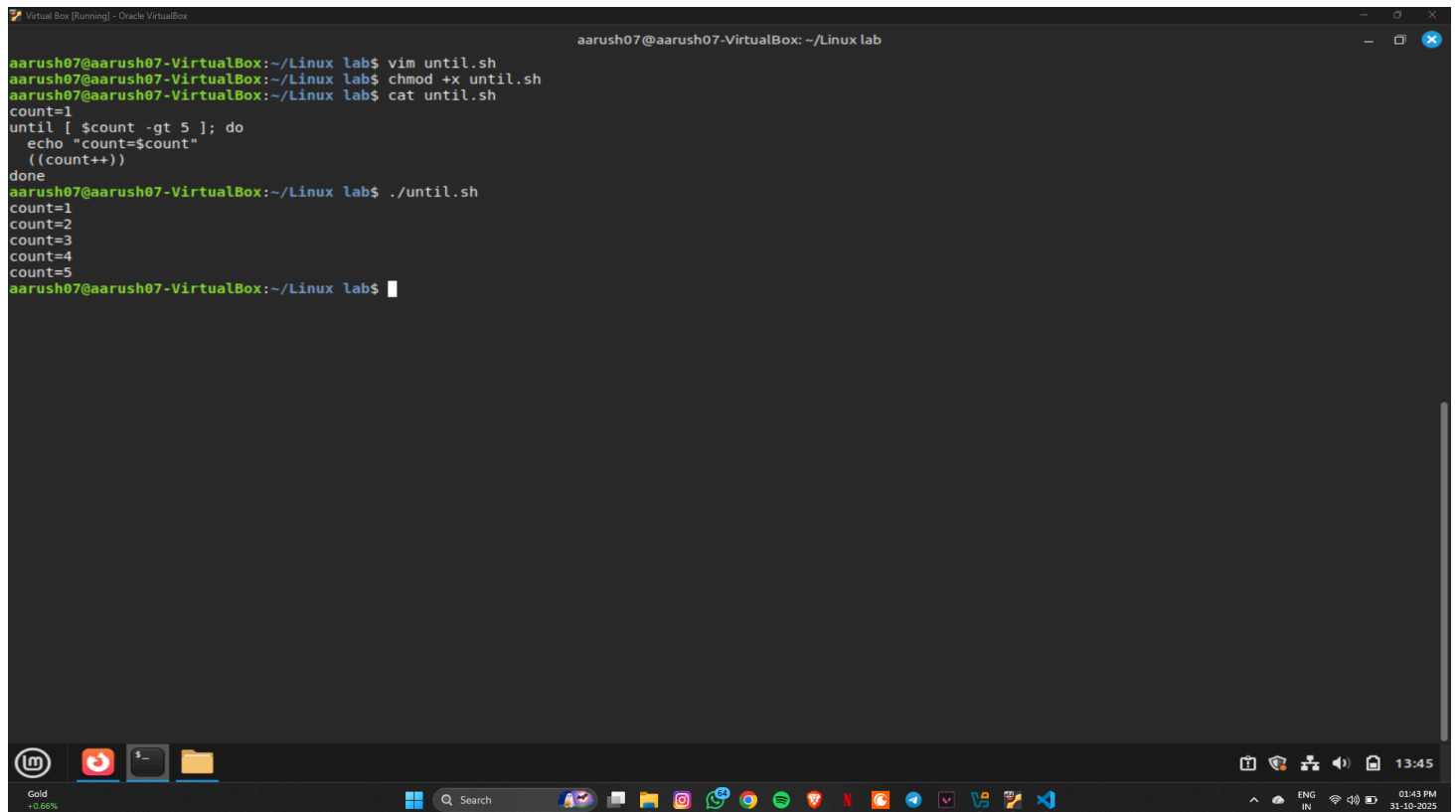
Task Statement:

Use an `until` loop to run until a condition becomes true.

Command(s):

```
count=1
until [ $count -gt 5 ]; do
    echo "count=$count"
    ((count++))
done
```

Output:



The screenshot shows a terminal window titled "Virtual Box (Running) - Oracle VM VirtualBox" with the command prompt "aarush07@aarush07-VirtualBox: ~/Linux lab". The user has created a script named "until.sh" and made it executable. The script contains an until loop that prints the value of "count" from 1 to 5. The output of the script is shown as follows:

```
aarush07@aarush07-VirtualBox:~/Linux lab$ vim until.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ chmod +x until.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ cat until.sh
count=1
until [ $count -gt 5 ]; do
    echo "count=$count"
    ((count++))
done
aarush07@aarush07-VirtualBox:~/Linux lab$ ./until.sh
count=1
count=2
count=3
count=4
count=5
aarush07@aarush07-VirtualBox:~/Linux lab$
```

The terminal window also shows a taskbar at the bottom with various application icons and system status indicators.

Exercise 6: break and continue

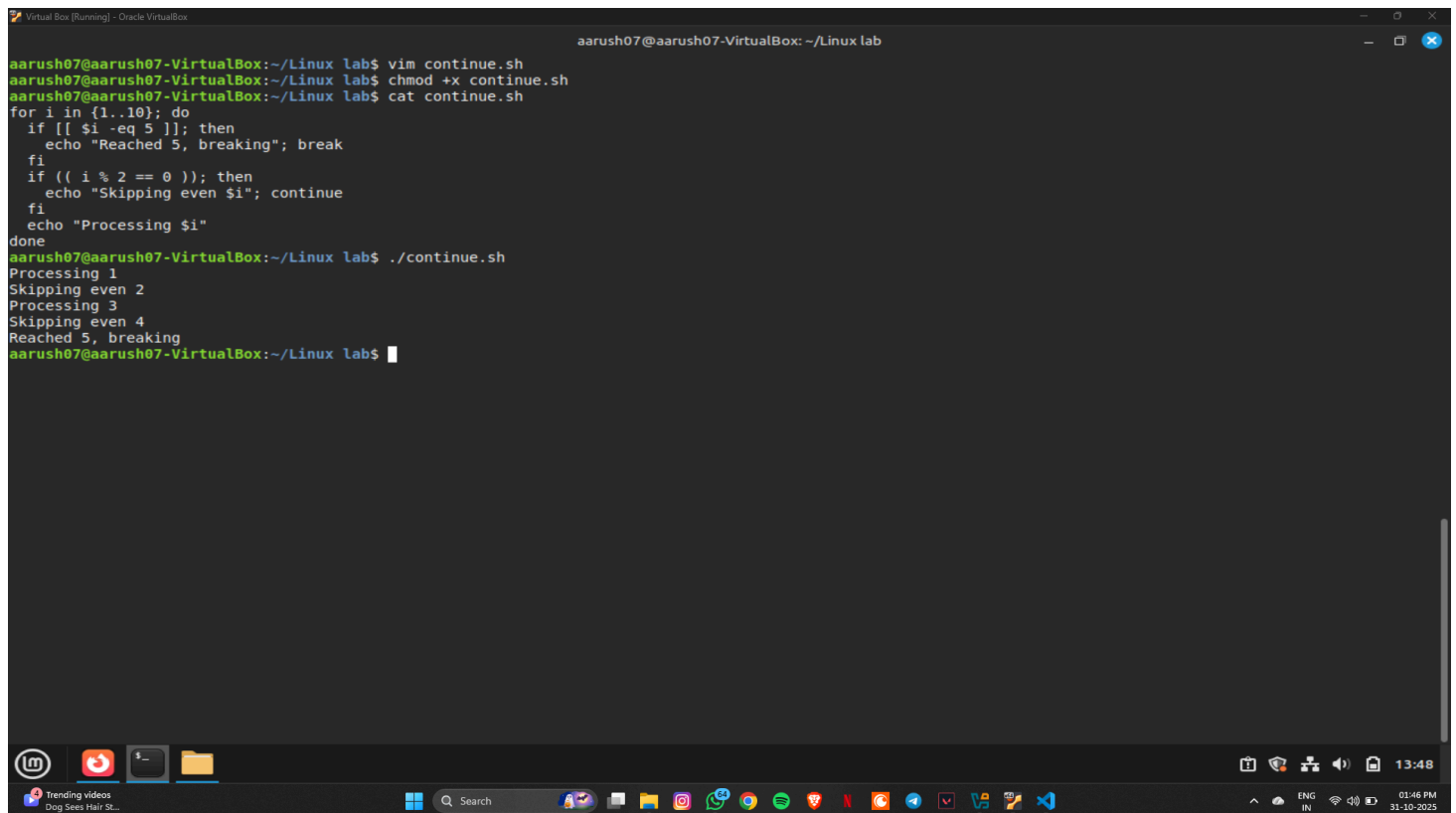
Task Statement:

Demonstrate break and continue inside a loop.

Command(s):

```
for i in {1..10}; do
    if [[ $i -eq 5 ]]; then
        echo "Reached 5, breaking"; break
    fi
    if (( i % 2 == 0 )); then
        echo "Skipping even $i"; continue
    fi
    echo "Processing $i"
done
```

Output:



```
aarush07@aarush07-VirtualBox: ~/Linux lab
aarush07@aarush07-VirtualBox:~/Linux lab$ vim continue.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ chmod +x continue.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ cat continue.sh
for i in {1..10}; do
    if [[ $i -eq 5 ]]; then
        echo "Reached 5, breaking"; break
    fi
    if (( i % 2 == 0 )); then
        echo "Skipping even $i"; continue
    fi
    echo "Processing $i"
done
aarush07@aarush07-VirtualBox:~/Linux lab$ ./continue.sh
Processing 1
Skipping even 2
Processing 3
Skipping even 4
Reached 5, breaking
aarush07@aarush07-VirtualBox:~/Linux lab$
```

Exercise 7: Nested loops

Task Statement:

Create nested loops to generate a multiplication table.

Command(s):

```
for i in {1..3}; do
  for j in {1..3}; do
    echo -n "${i}*j) "
  done
done
```

Output:



Result

- Implemented `for`, `while`, and `until` loops and used loop control statements.
- Practiced reading input, processing files, and nested iteration.

Challenges Faced & Learning Outcomes

- Challenge 1: Handling spaces and special characters when iterating filenames — learned to use quotes and `read -r`.
- Challenge 2: Remembering arithmetic syntax in Bash — used `(())` and `expr` where needed.

Learning:

- Loops are powerful for automation in shell scripting. Correct quoting and use of control constructs prevent common bugs.

Conclusion

The lab demonstrated practical loop constructs in Bash for automating repetitive tasks and processing data efficiently.