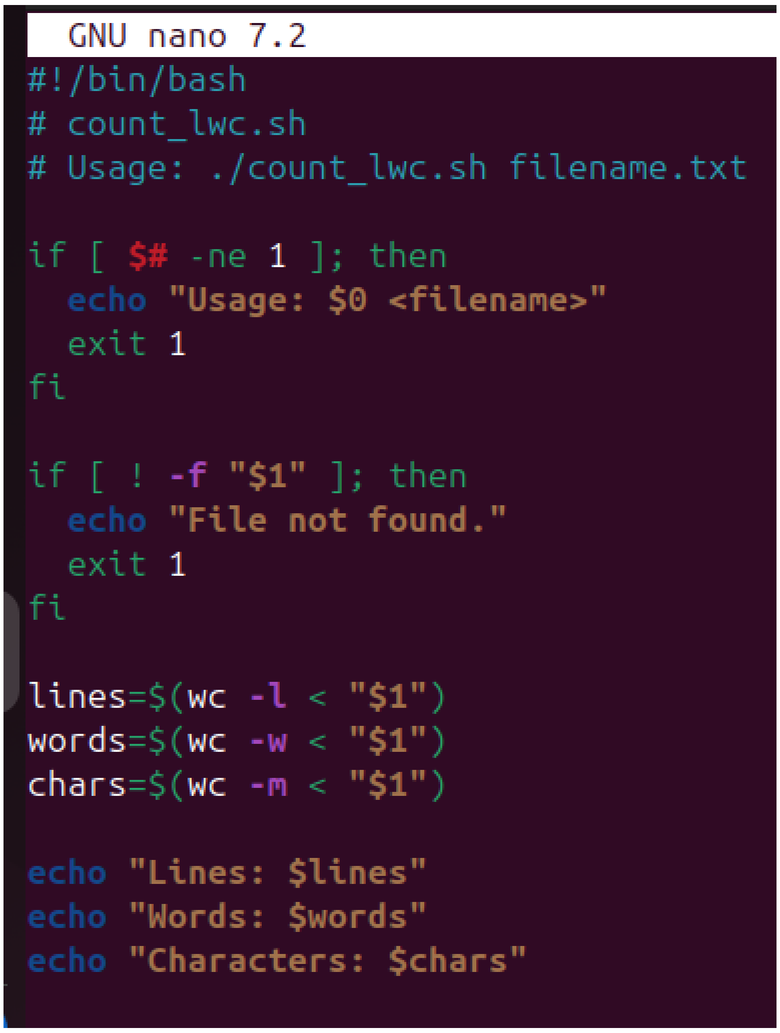
Bash Script to Count Lines, Words, and Characters in a File

Bash Script to Count Lines, Words, and Characters in a File

Code



```
  GNU nano 7.2
#!/bin/bash
# count_lwc.sh
# Usage: ./count_lwc.sh filename.txt

if [ $# -ne 1 ]; then
  echo "Usage: $0 <filename>"
  exit 1
fi

if [ ! -f "$1" ]; then
  echo "File not found."
  exit 1
fi

lines=$(wc -l < "$1")
words=$(wc -w < "$1")
chars=$(wc -m < "$1")

echo "Lines: $lines"
echo "Words: $words"
echo "Characters: $chars"
```

Line by line explanation

```
#!/bin/bash
```

This is called a shebang (#!).

It tells the system which interpreter to use to run this script.

In this case, /bin/bash means the script should be run using Bash, which is a popular shell on Unix-like systems.

```
# count_lwc.sh
```

This is a comment.

It’s just documentation — not executed.

Tells the reader the script name.

```
# Usage: ./count_lwc.sh filename.txt
```

Another comment.

Shows how to use the script properly.

./count_lwc.sh means run the script, and filename.txt is the input file.

📌 This is helpful for users and future developers.

```
if [ $# -ne 1 ]; then
  echo "Usage: $0 <filename>"
  exit 1
fi
```

if [$# -ne 1]:

$# = number of arguments passed to the script.

-ne = “not equal”

So this checks: Did the user pass exactly 1 argument?

If not, then:

echo "Usage: $0 ":

Shows how to use the script.

$0 = name of the script (e.g., count_lwc.sh)

exit 1:

Stops the script and returns an error code 1.

📌 Why is this important?
Ensures the script is used correctly. Avoids confusion or errors from missing arguments.

```
if [ ! -f "$1" ]; then
  echo "File not found."
  exit 1
fi
```

"$1" = the first argument (should be the filename).

-f "$1" = checks if a file exists and is a regular file.

! -f = "file does not exist"

So this checks:

❗ If the file does not exist, then print an error and exit.

📌 Why is this important?
It prevents the script from trying to read a file that doesn’t exist, which would cause an error.

```
lines=$(wc -l < "$1")
```

Breakdown:

wc -l < "$1":

wc = word count command

-l = count lines

< "$1" = redirect the contents of the file into the command

$(...) = command substitution: runs the command and stores its result

lines=... = save the result in the lines variable

📌 This command stores the number of lines in the file into the variable lines.

```
words=$(wc -w < "$1")
```

Breakdown:

wc -w = counts words

Stores the result in the variable words

📌 This gives you the word count of the file.

```
chars=$(wc -m < "$1")
```

Breakdown:

wc -m = counts characters

Stores the result in chars

📌 Counts all characters, including spaces and newline characters.

```
echo "Lines: $lines"
```

echo prints text to the terminal.

$lines is replaced by the value stored earlier.

📌 Prints the number of lines in the file.

```
echo "Words: $words"
```

📌 Prints the number of words in the file

```
echo "Characters: $chars"
```

📌 Prints the number of characters in the file.

Ques : What This Script Does ?

Answer:

> Checks if exactly one argument is given.

> Validates that the file exists.

> Uses wc to:

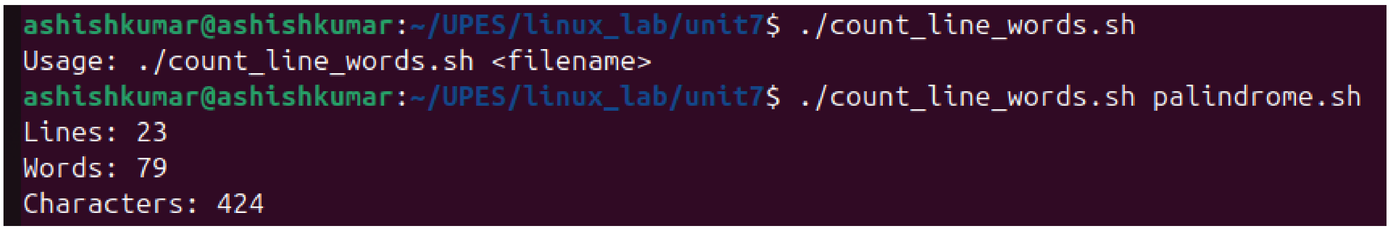
> Count lines

> Count words

> Count characters

> Prints the counts in a readable format.

For example :



```
ashishkumar@ashishkumar:~/UPES/linux_lab/unit7$ ./count_line_words.sh
Usage: ./count_line_words.sh <filename>
ashishkumar@ashishkumar:~/UPES/linux_lab/unit7$ ./count_line_words.sh palindrome.sh
Lines: 23
Words: 79
Characters: 424
```