Bash Project

Bash Case

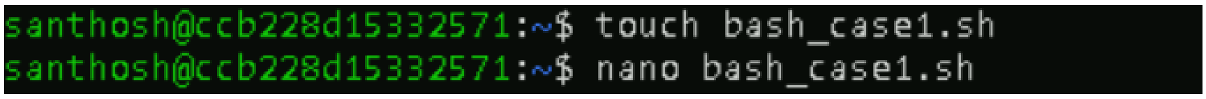
The **Bash case** statement is the simplest form of IF-THEN-ELSE with many ELIF elements. Using the case statement makes our bash script more readable and easier to maintain. These are generally applied to simplify the complex conditions having multiple different choices.

The Bash case statement follows a similar logic as the Javascript or C switch statement. There is a slight difference, as follows:

o The Bash case statement takes a value once and tests that value multiple times. It stops searching for a pattern once it has found it and executed the statement linked with it, which is almost opposite in case of the C switch statement.

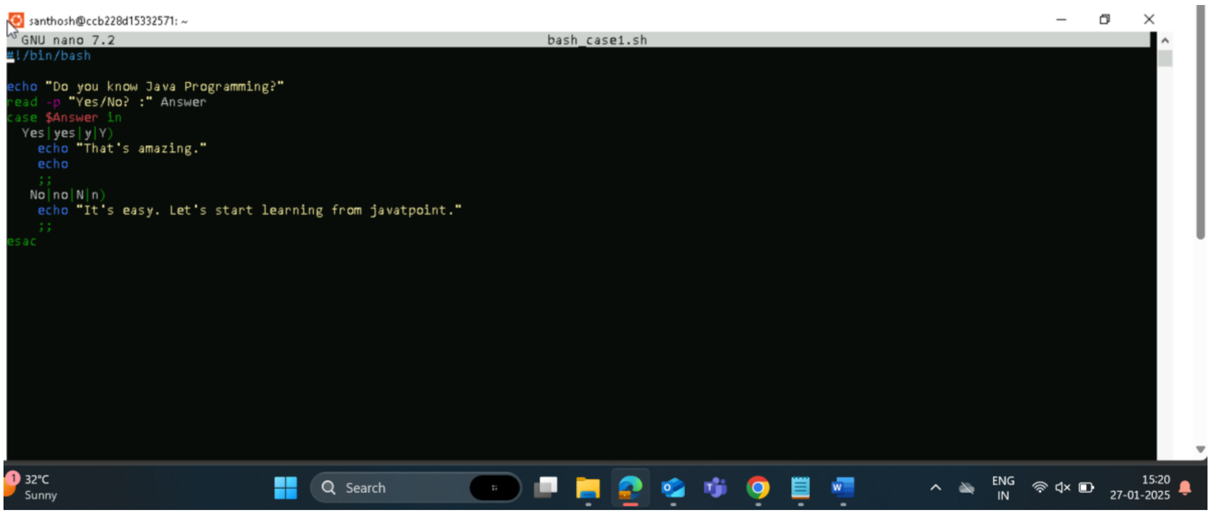
Example 1

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



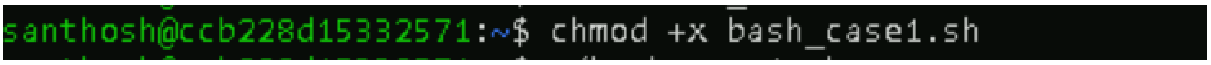
```
santhosh@ccb228d15332571:~$ touch bash_case1.sh
santhosh@ccb228d15332571:~$ nano bash_case1.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                              bash_case1.sh
#!/bin/bash

echo "Do you know Java Programming?"
read -p "Yes/No? :" Answer
case $Answer in
  Yes|yes|y|Y)
    echo "That's amazing."
    echo
    ;;
   No|no|N|n)
    echo "It's easy. Let's start learning from javatpoint."
    ;;
esac
```

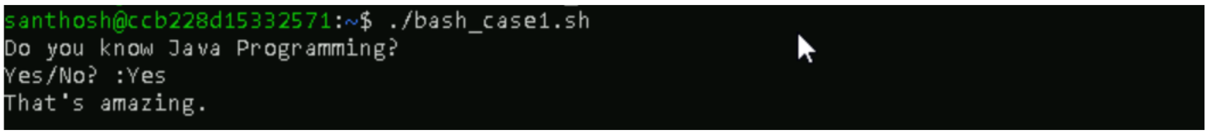
Step 3: Providing the necessary permissions for the base_case1.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x bash_case1.sh
```

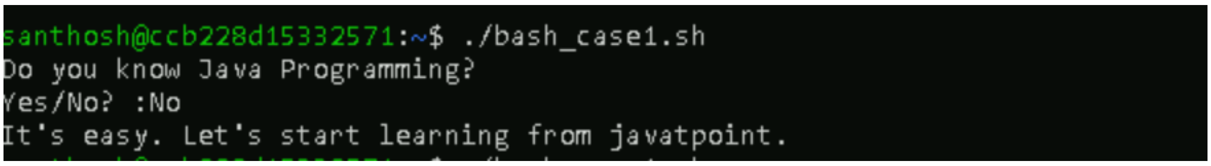
Step 4: Executing the output.

a. For Yes the output is.



```
santhosh@ccb228d15332571:~$ ./bash_case1.sh
Do you know Java Programming?
Yes/No? :Yes
That's amazing.
```

b. For No the output is.

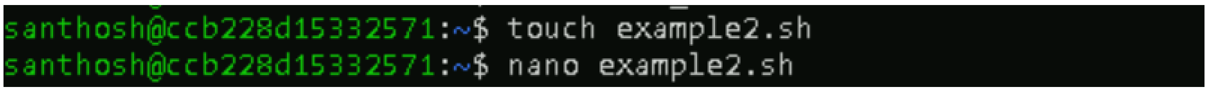


```
santhosh@ccb228d15332571:~$ ./bash_case1.sh
Do you know Java Programming?
Yes/No? :No
It's easy. Let's start learning from javatpoint.
```

Example 2

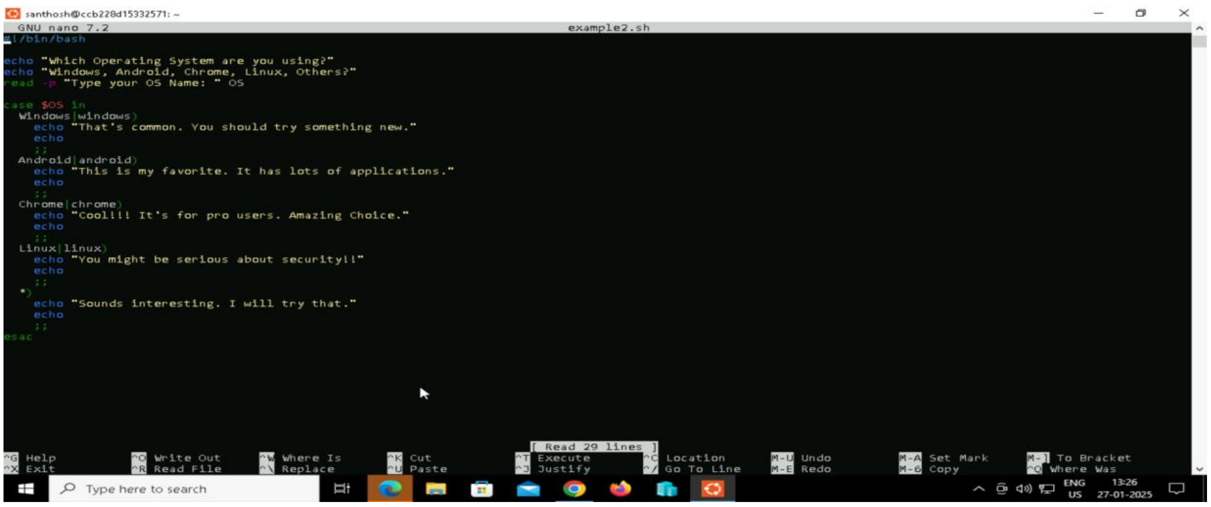
A combined scenario where there is also a default case when no previous matched case is found.

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



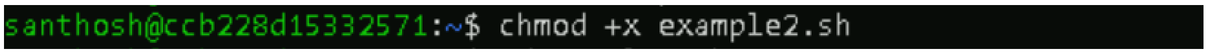
```
santhosh@ccb228d15332571:~$ touch example2.sh
santhosh@ccb228d15332571:~$ nano example2.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



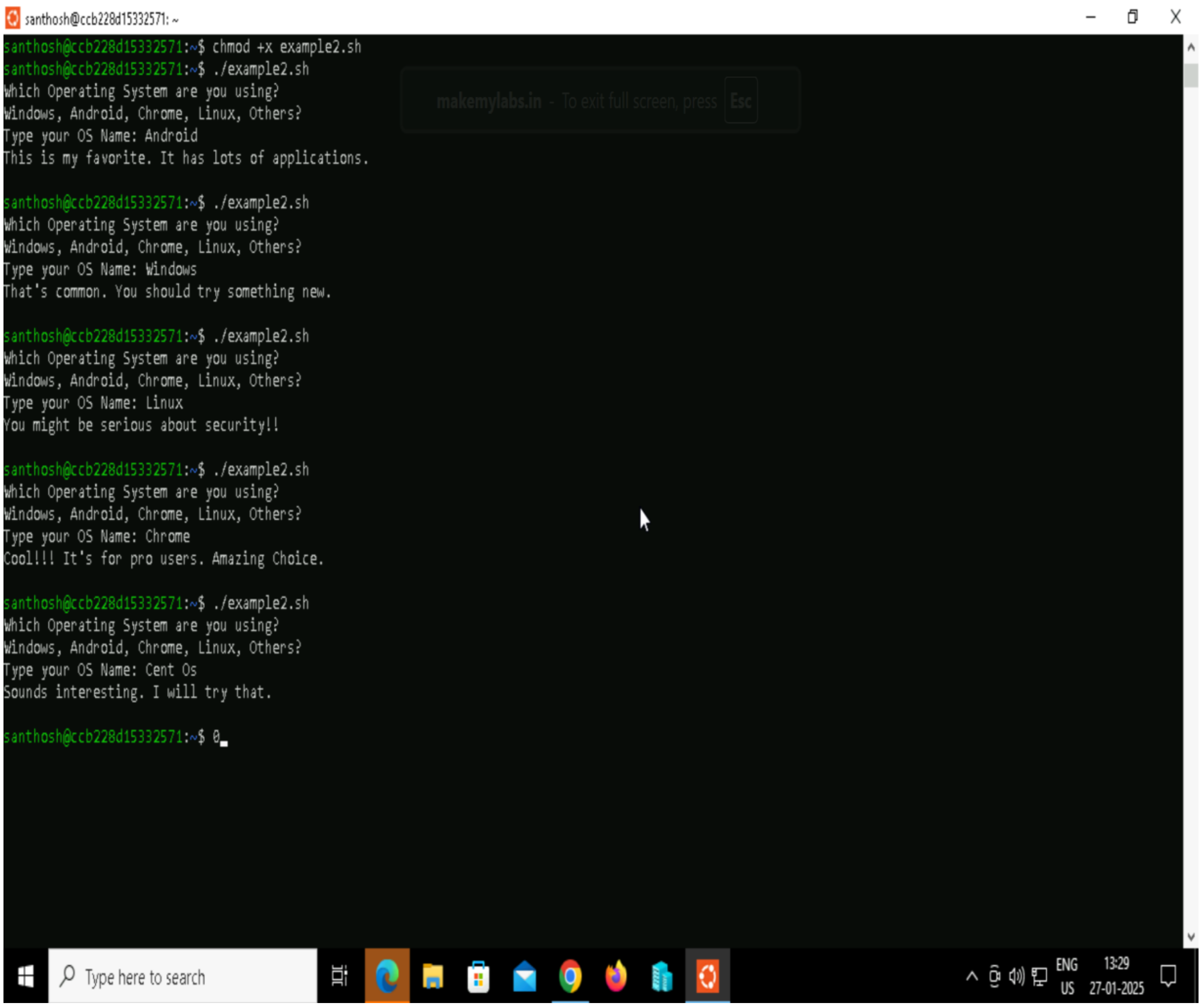
```
#!/bin/bash

echo "Which Operating System are you using?"
echo "Windows, Android, Chrome, Linux, Others?"
read -p "Type your OS Name: " OS

case $OS in
  Windows|windows)
    echo "That's common. You should try something new."
    echo
    ;;
  Android|android)
    echo "This is my favorite. It has lots of applications."
    echo
    ;;
  Chrome|chrome)
    echo "Cool!!! It's for pro users. Amazing Choice."
    echo
    ;;
  Linux|linux)
    echo "You might be serious about security!!"
    echo
    ;;
  *)
    echo "Sounds interesting. I will try that."
    echo
    ;;
esac
```

Step 3: Providing the necessary permissions for the example2.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x example2.sh
```

Step 4: Executing the output.


santhosh@ccb228d15332571: ~
makemylabs.in - To exit full screen, press Esc
Type here to search
ENG
US
13:29
27-01-2025

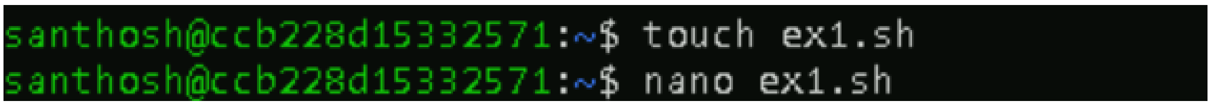
```
santhosh@ccb228d15332571:~$ chmod +x example2.sh
santhosh@ccb228d15332571:~$ ./example2.sh
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name: Android
This is my favorite. It has lots of applications.

santhosh@ccb228d15332571:~$ ./example2.sh
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name: Windows
That's common. You should try something new.

santhosh@ccb228d15332571:~$ ./example2.sh
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name: Linux
You might be serious about security!!

santhosh@ccb228d15332571:~$ ./example2.sh
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name: Chrome
Cool!!! It's for pro users. Amazing Choice.

santhosh@ccb228d15332571:~$ ./example2.sh
Which Operating System are you using?
Windows, Android, Chrome, Linux, Others?
Type your OS Name: Cent Os
Sounds interesting. I will try that.

santhosh@ccb228d15332571:~$ @
```

Bash For Loop

Like any other programming language, bash shell scripting also supports 'for loops' to perform repetitive tasks. It helps us to iterate a particular set of statements over a series of words in a string, or elements in an array. For example, you can either run UNIX command (or task) many times or just read and process the list of commands using a 'for loop'.

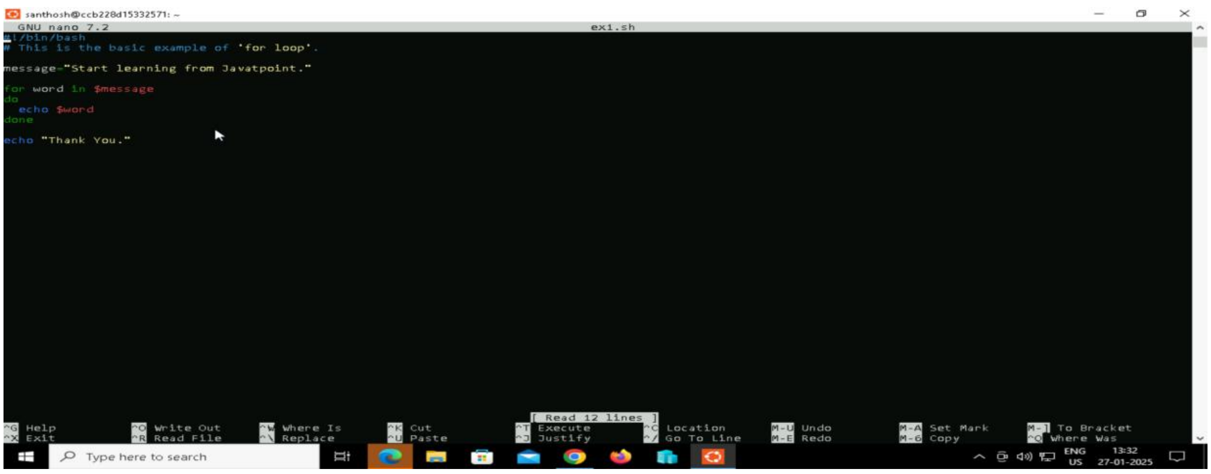
Example 1

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



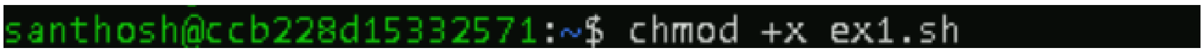
```
santhosh@ccb228d15332571:~$ touch ex1.sh
santhosh@ccb228d15332571:~$ nano ex1.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



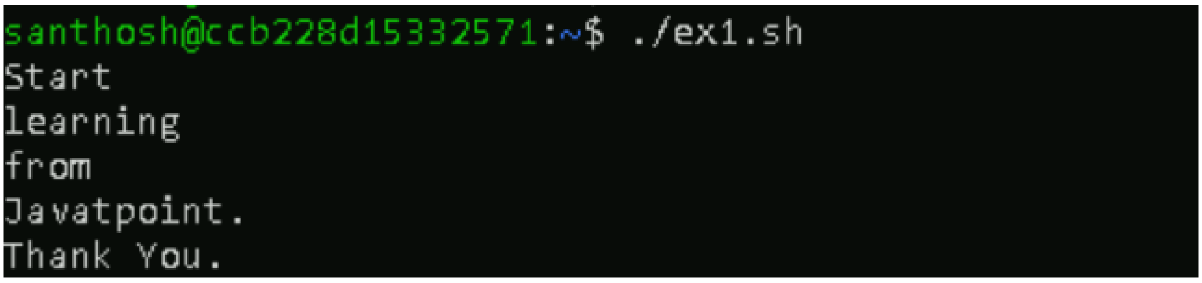
```
#!/bin/bash
# This is the basic example of 'for loop'.

message="Start learning from Javatpoint."

for word in $message
do
  echo $word
done

echo "Thank You."
```

Step 3: Providing the necessary permissions for the ex1.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x ex1.sh
```

Step 4: Executing the output.

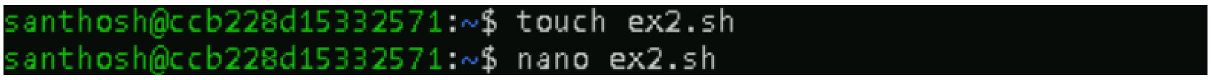


```
santhosh@ccb228d15332571:~$ ./ex1.sh
Start
learning
from
Javatpoint.
Thank You.
```

Example 2

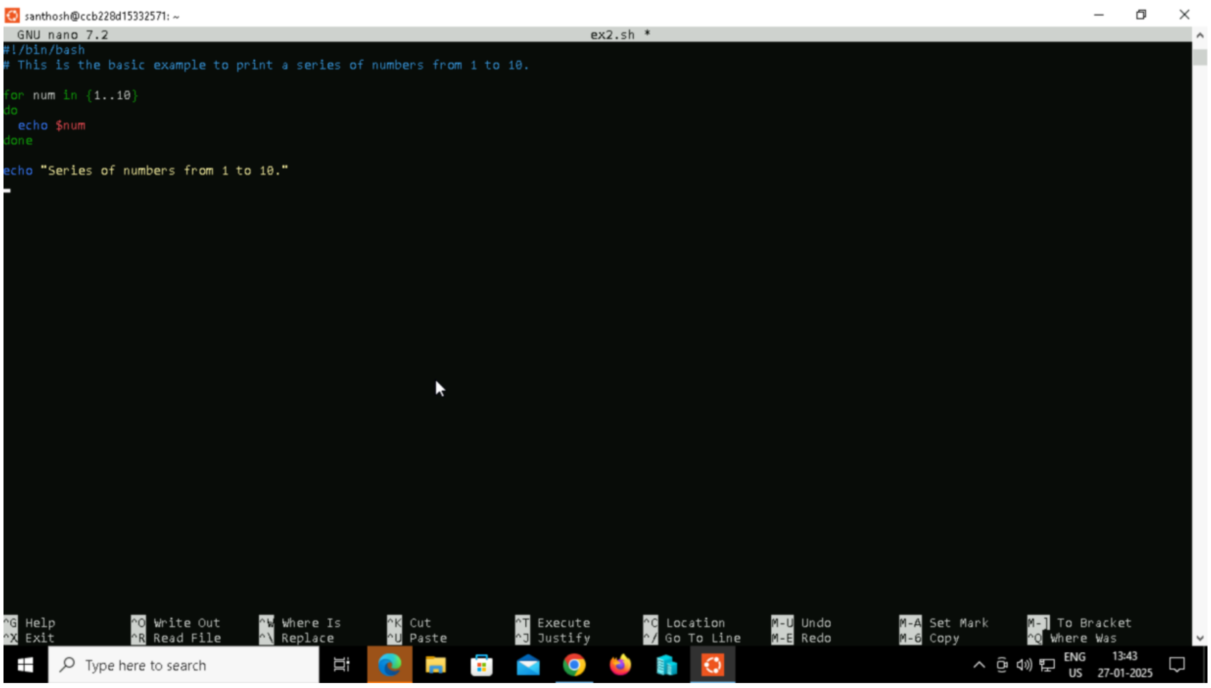
For Loop to Read a Range

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



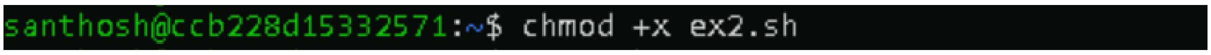
```
santhosh@ccb228d15332571:~$ touch ex2.sh
santhosh@ccb228d15332571:~$ nano ex2.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



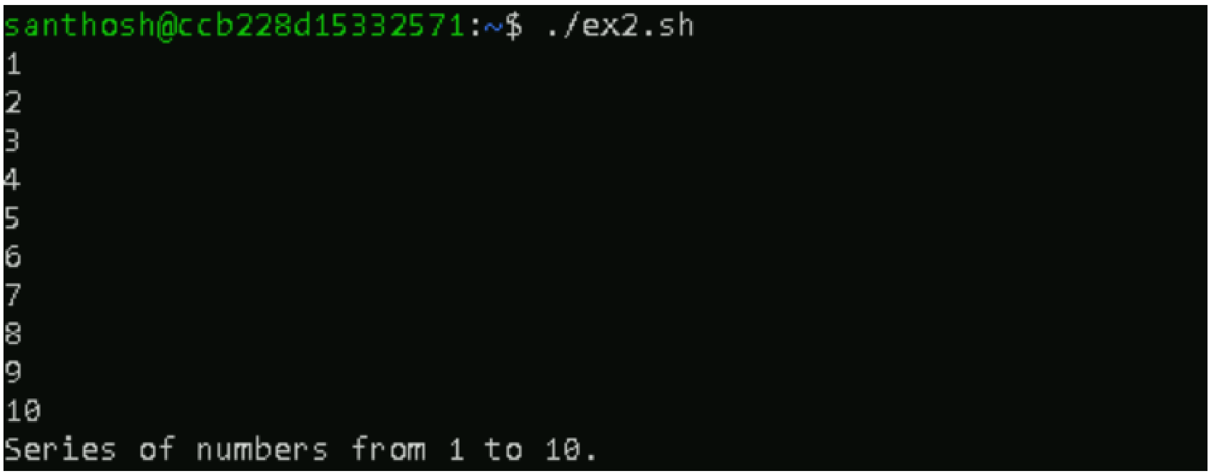
```
  GNU nano 7.2                                   ex2.sh *
#!/bin/bash
# This is the basic example to print a series of numbers from 1 to 10.

for num in {1..10}
do
  echo $num
done

echo "Series of numbers from 1 to 10."
```

Step 3: Providing the necessary permissions for the ex2.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x ex2.sh
```

Step 4: Executing the output.

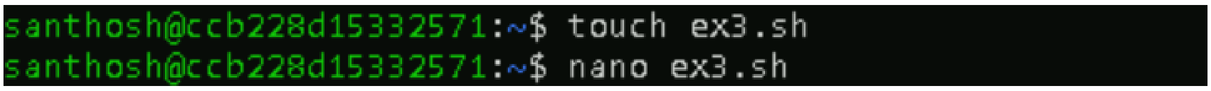


```
santhosh@ccb228d15332571:~$ ./ex2.sh
1
2
3
4
5
6
7
8
9
10
Series of numbers from 1 to 10.
```

Example 3

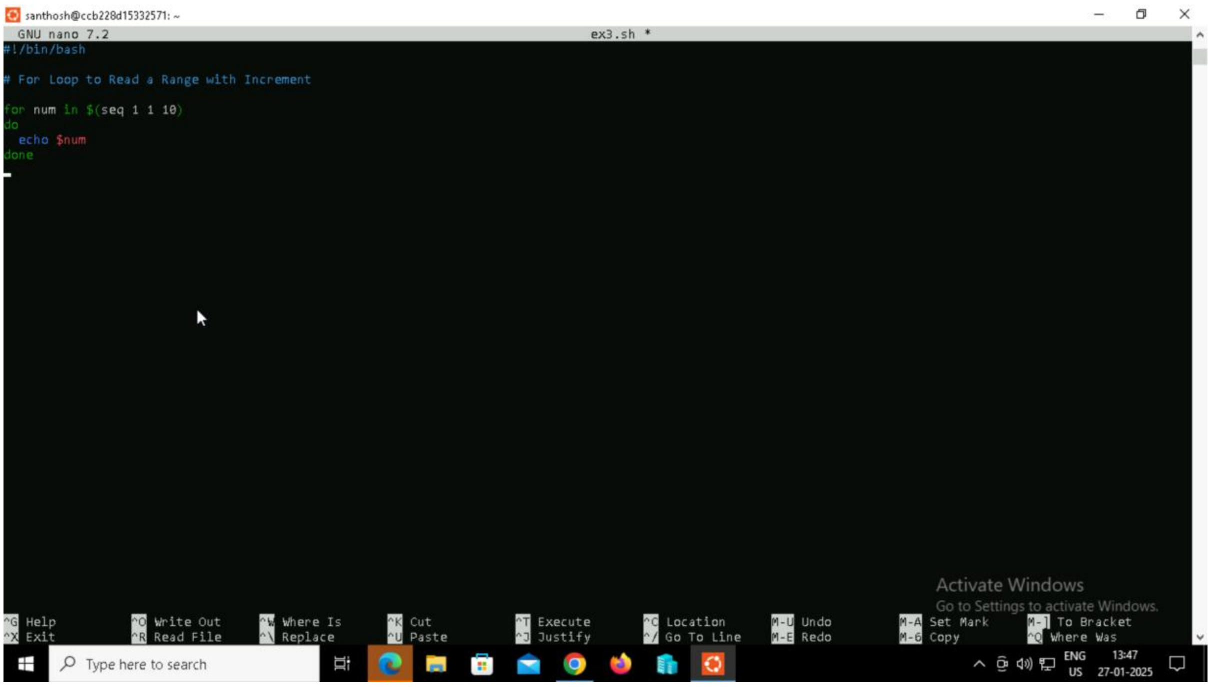
For Loop to Read a Range with Increment/Decrement

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



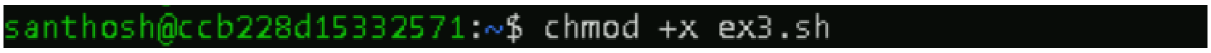
```
santhosh@ccb228d15332571:~$ touch ex3.sh
santhosh@ccb228d15332571:~$ nano ex3.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



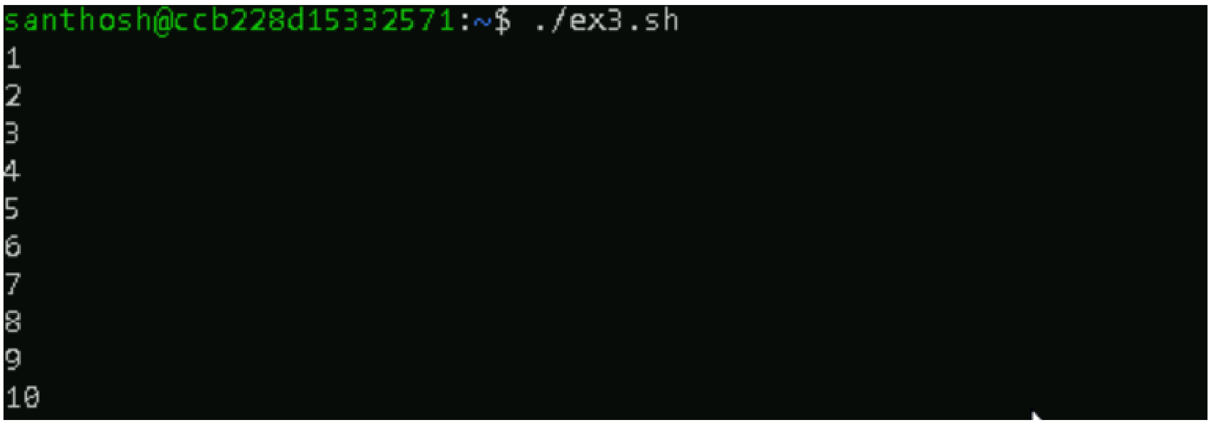
```
  GNU nano 7.2                                    ex3.sh *
#!/bin/bash

# For Loop to Read a Range with Increment

for num in $(seq 1 1 10)
do
  echo $num
done
```

Step 3: Providing the necessary permissions for the ex3.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x ex3.sh
```

Step 4: Executing the output.

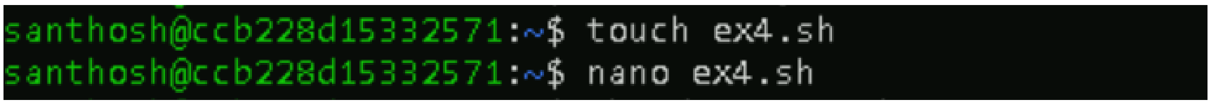


```
santhosh@ccb228d15332571:~$ ./ex3.sh
1
2
3
4
5
6
7
8
9
10
```

Example 4

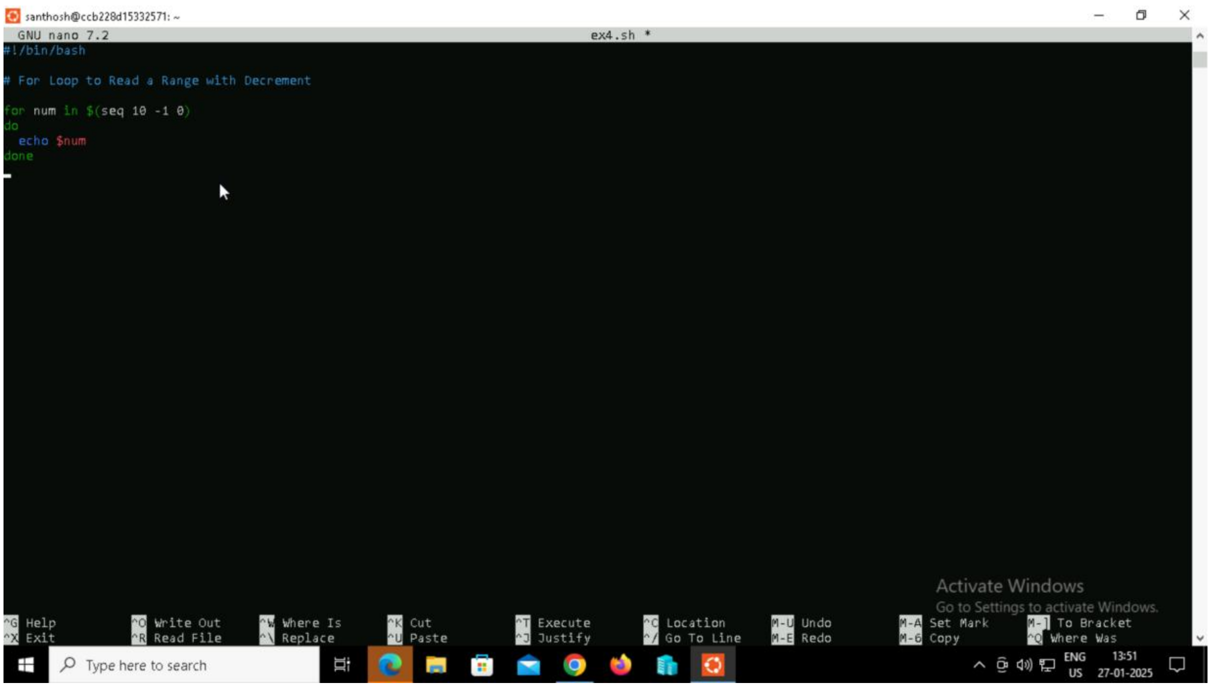
For Decrement

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



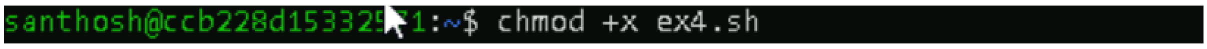
```
santhosh@ccb228d15332571:~$ touch ex4.sh
santhosh@ccb228d15332571:~$ nano ex4.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



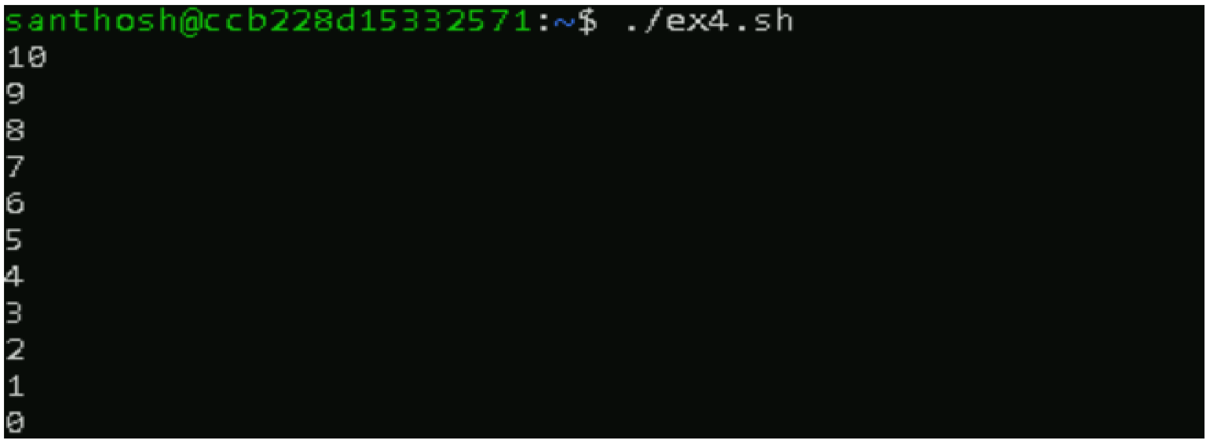
```
GNU nano 7.2                              ex4.sh *
#!/bin/bash

# For Loop to Read a Range with Decrement

for num in $(seq 10 -1 0)
do
  echo $num
done
```

Step 3: Providing the necessary permissions for the ex4.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x ex4.sh
```

Step 4: Executing the output.

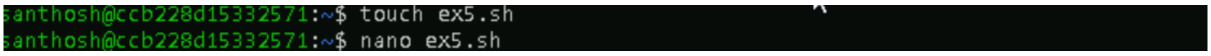


```
santhosh@ccb228d15332571:~$ ./ex4.sh
10
9
8
7
6
5
4
3
2
1
0
```

Example 5

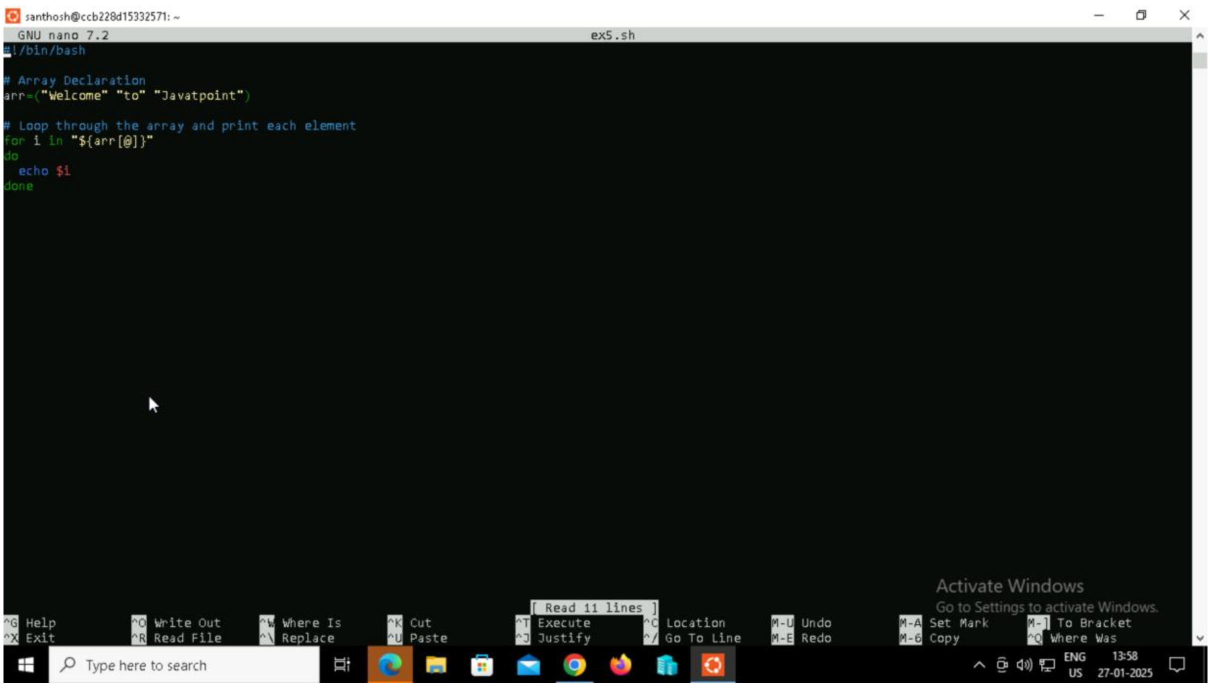
For Loop to Read Array Variables

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



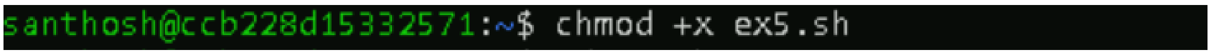
```
santhosh@ccb228d15332571:~$ touch ex5.sh
santhosh@ccb228d15332571:~$ nano ex5.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement**.**



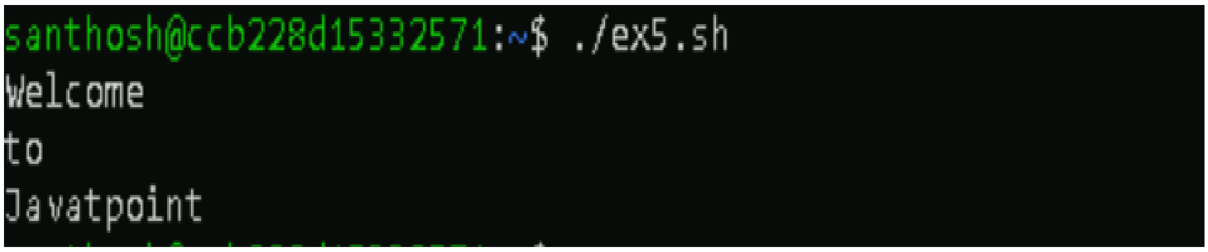
```
  GNU nano 7.2                                ex5.sh
#!/bin/bash

# Array Declaration
arr=("Welcome" "to" "Javatpoint")

# Loop through the array and print each element
for i in "${arr[@]}"
do
  echo $i
done
```

Step 3: Providing the necessary permissions for the ex5.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x ex5.sh
```

Step 4: Executing the output.

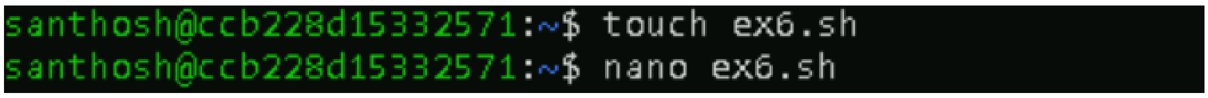


```
santhosh@ccb228d15332571:~$ ./ex5.sh
Welcome
to
Javatpoint
```

Example 6

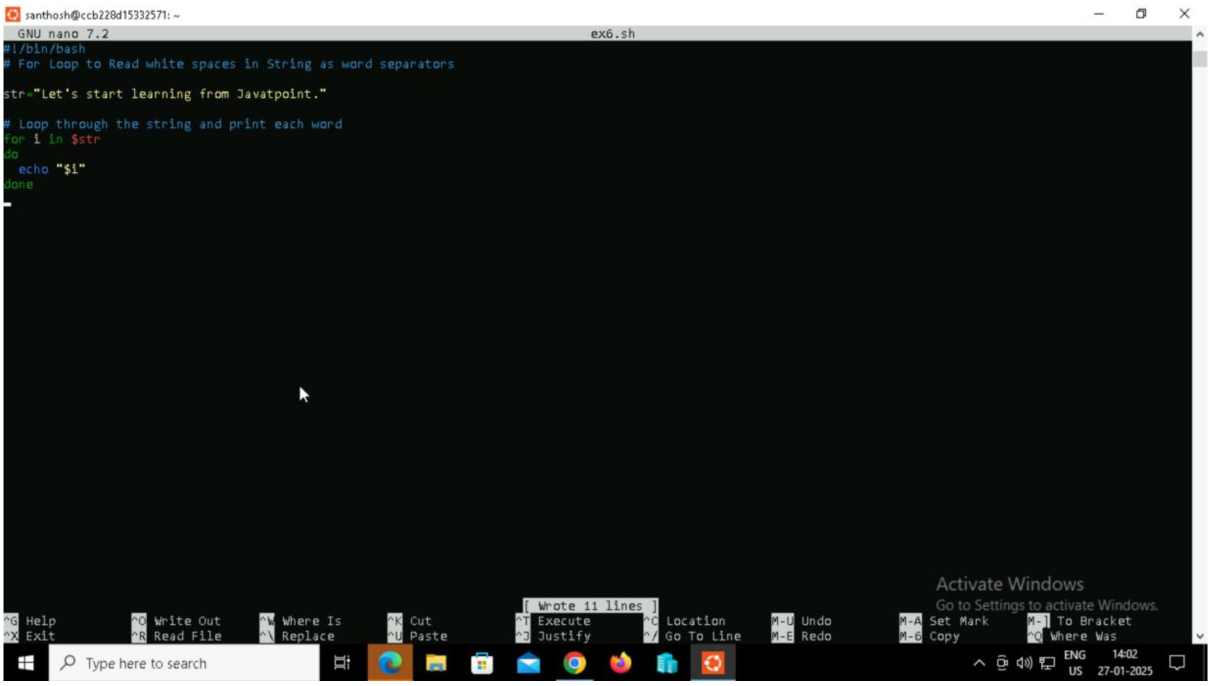
For Loop to Read white spaces in String as word separators.

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



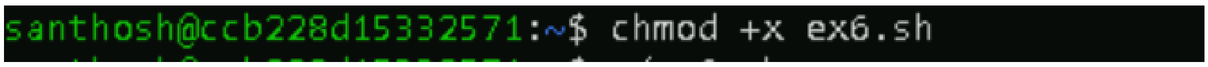
```
santhosh@ccb228d15332571:~$ touch ex6.sh
santhosh@ccb228d15332571:~$ nano ex6.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



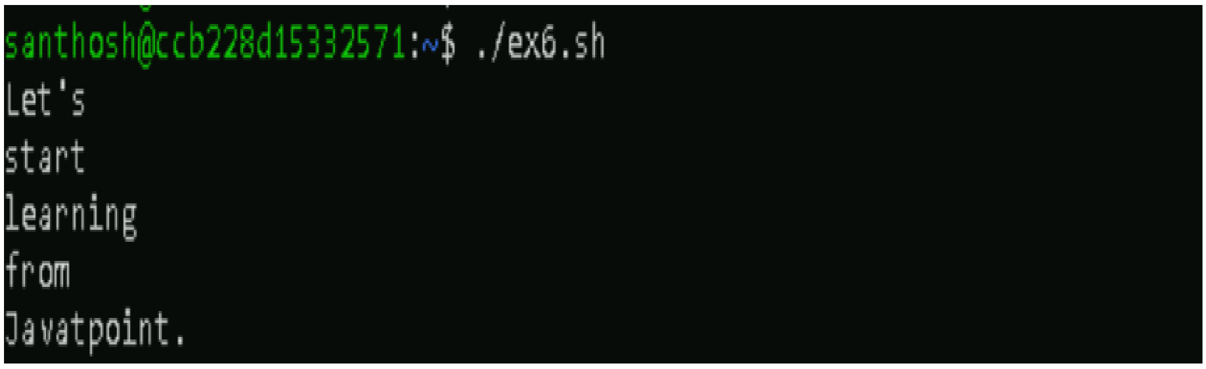
```
GNU nano 7.2                         ex6.sh
#!/bin/bash
# For Loop to Read white spaces in String as word separators

str="Let's start learning from Javatpoint."

# Loop through the string and print each word
for i in $str
do
  echo "$i"
done

                              [ Wrote 11 lines ]
```

Step 3: Providing the necessary permissions for the ex6.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x ex6.sh
```

Step 4: Executing the output.

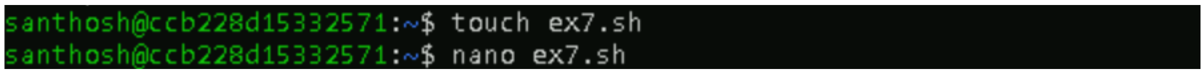


```
santhosh@ccb228d15332571:~$ ./ex6.sh
Let's
start
learning
from
Javatpoint.
```

Example 7

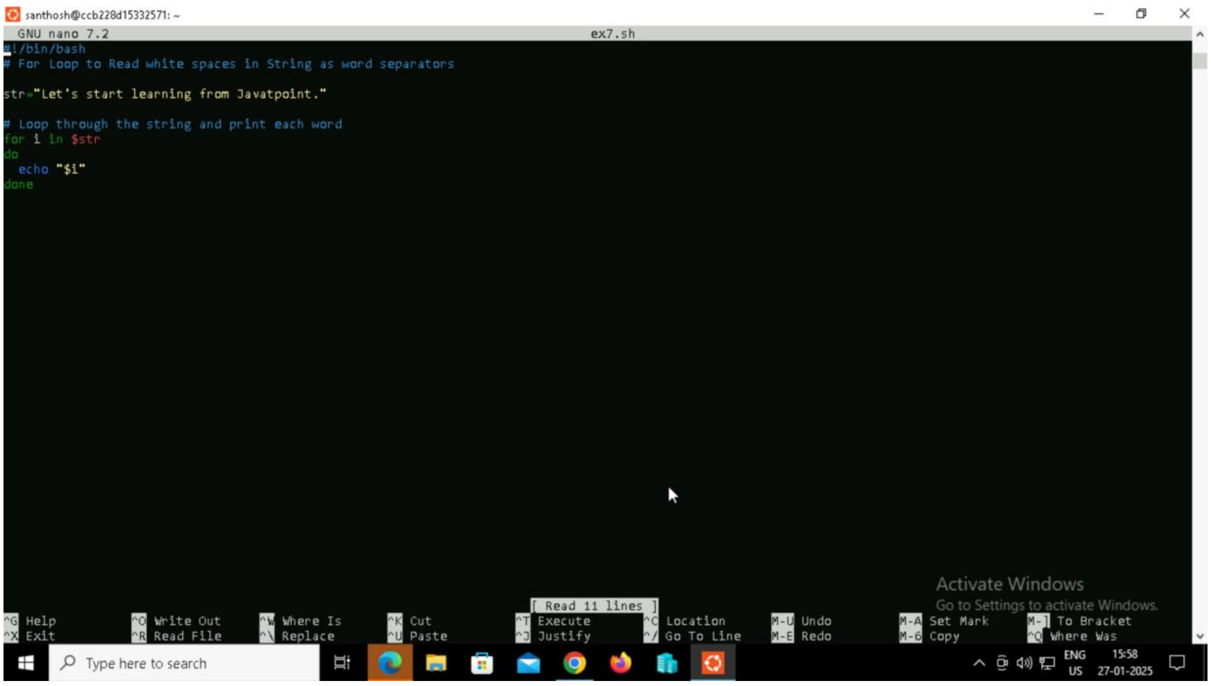
For Loop to Read each line in String as a word.

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



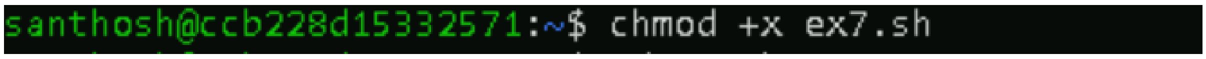
```
santhosh@ccb228d15332571:~$ touch ex7.sh
santhosh@ccb228d15332571:~$ nano ex7.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



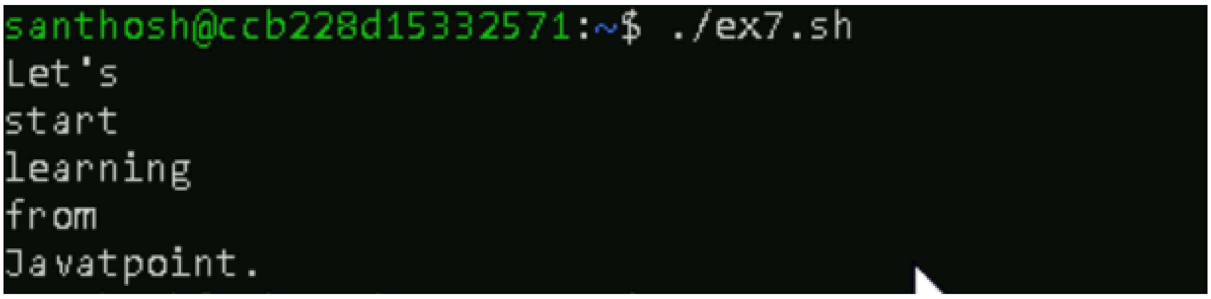
```
  GNU nano 7.2                                ex7.sh
#!/bin/bash
# For Loop to Read white spaces in String as word separators

str="Let's start learning from Javatpoint."

# Loop through the string and print each word
for i in $str
do
  echo "$i"
done
```

Step 3: Providing the necessary permissions for the ex7.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x ex7.sh
```

Step 4: Executing the output.

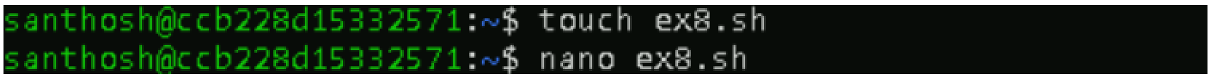


```
santhosh@ccb228d15332571:~$ ./ex7.sh
Let's
start
learning
from
Javatpoint.
```

Example 8

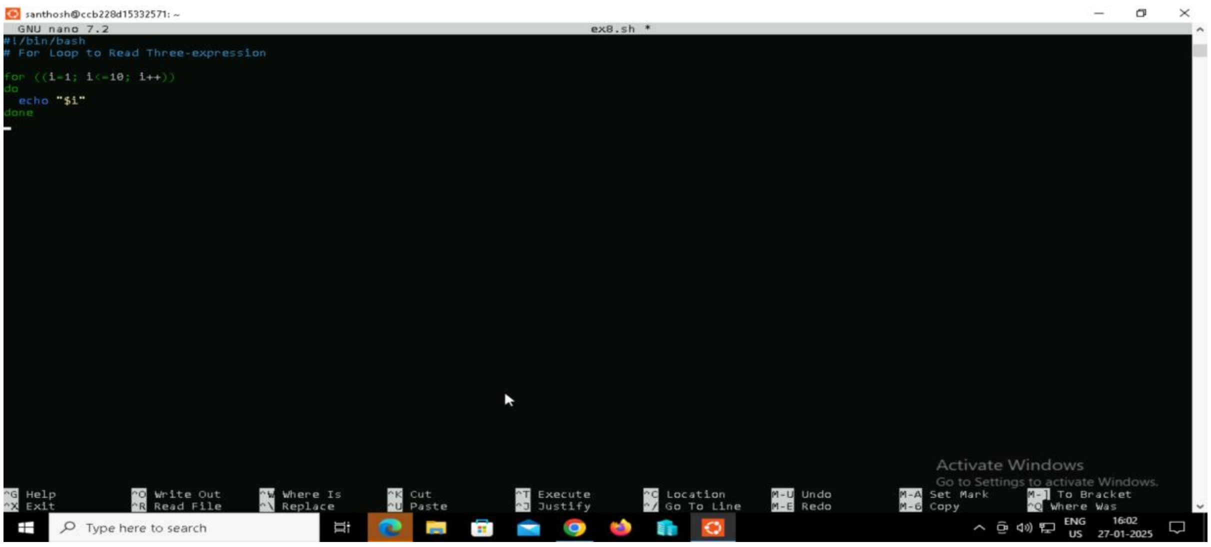
For Loop to Read Three-expression

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



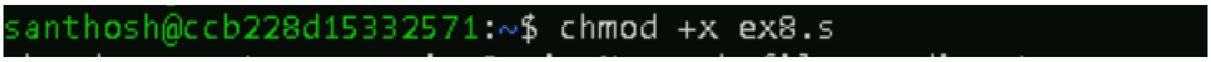
```
santhosh@ccb228d15332571:~$ touch ex8.sh
santhosh@ccb228d15332571:~$ nano ex8.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



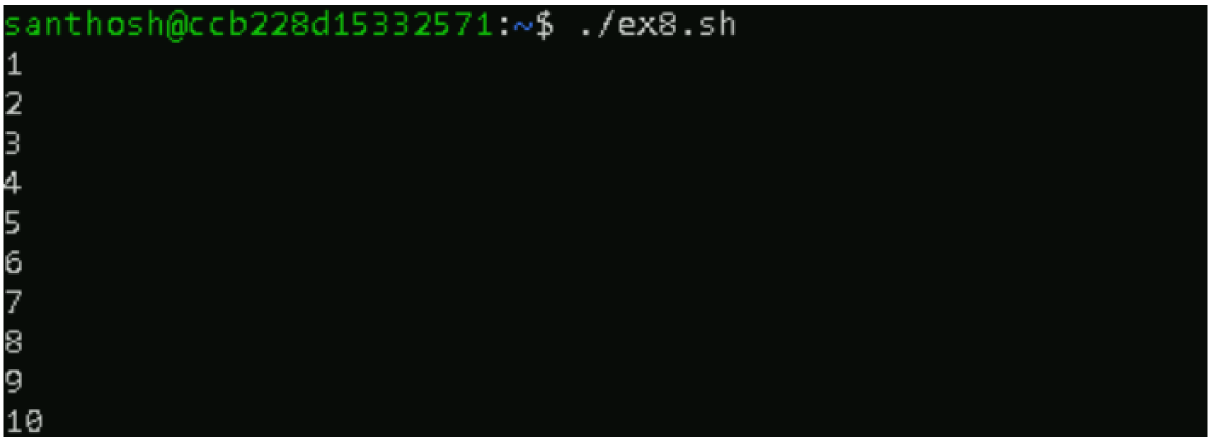
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                                ex8.sh *
#!/bin/bash
# For Loop to Read Three-expression

for ((i=1; i<=10; i++))
do
  echo "$i"
done
```

Step 3: Providing the necessary permissions for the ex8.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x ex8.s
```

Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./ex8.sh
1
2
3
4
5
6
7
8
9
10
```

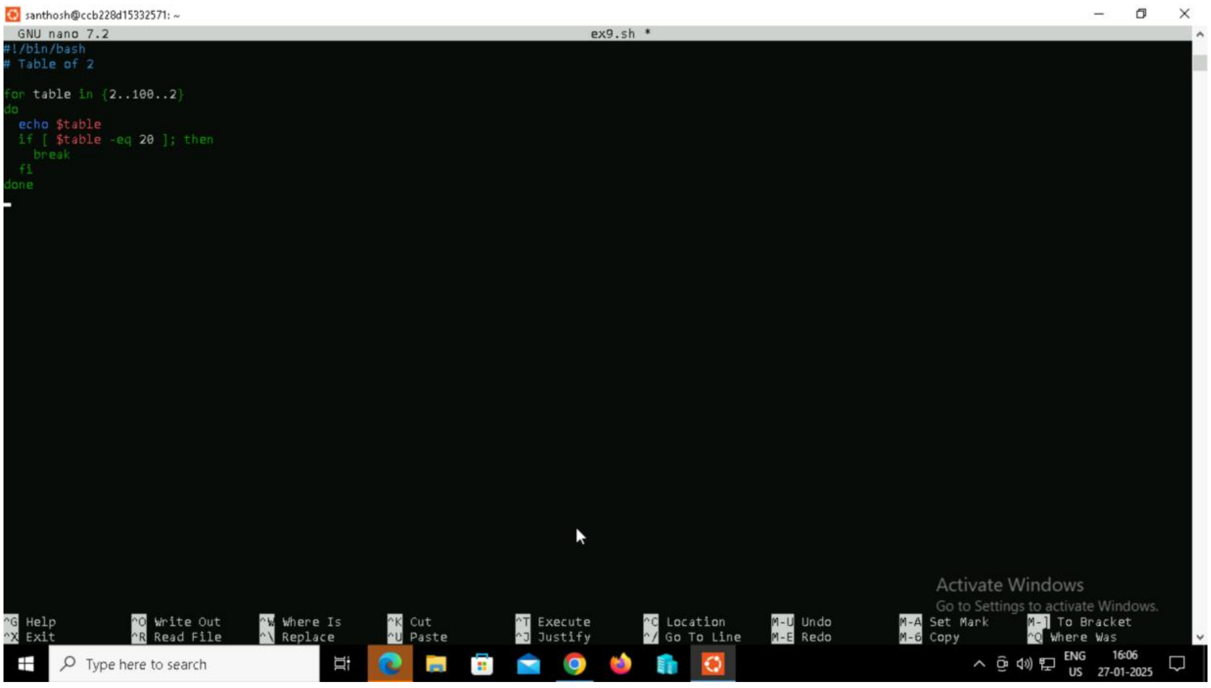
Example 9

For Loop with a Break Statement

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.

```
santhosh@ccb228d15332571:~$ touch ex9.sh
santhosh@ccb228d15332571:~$ nano ex9.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



```
GNU nano 7.2                         ex9.sh *
#!/bin/bash
# Table of 2

for table in {2..100..2}
do
  echo $table
  if [ $table -eq 20 ]; then
    break
  fi
done
```

Step 3: Providing the necessary permissions for the ex.sh script.

```
santhosh@ccb228d15332571:~$ chmod +x ex9.sh
```

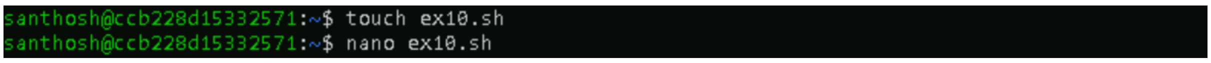
Step 4: Executing the output.

```
santhosh@ccb228d15332571:~$ ./ex.sh
2
4
6
8
10
12
14
16
18
20
```

Example 10

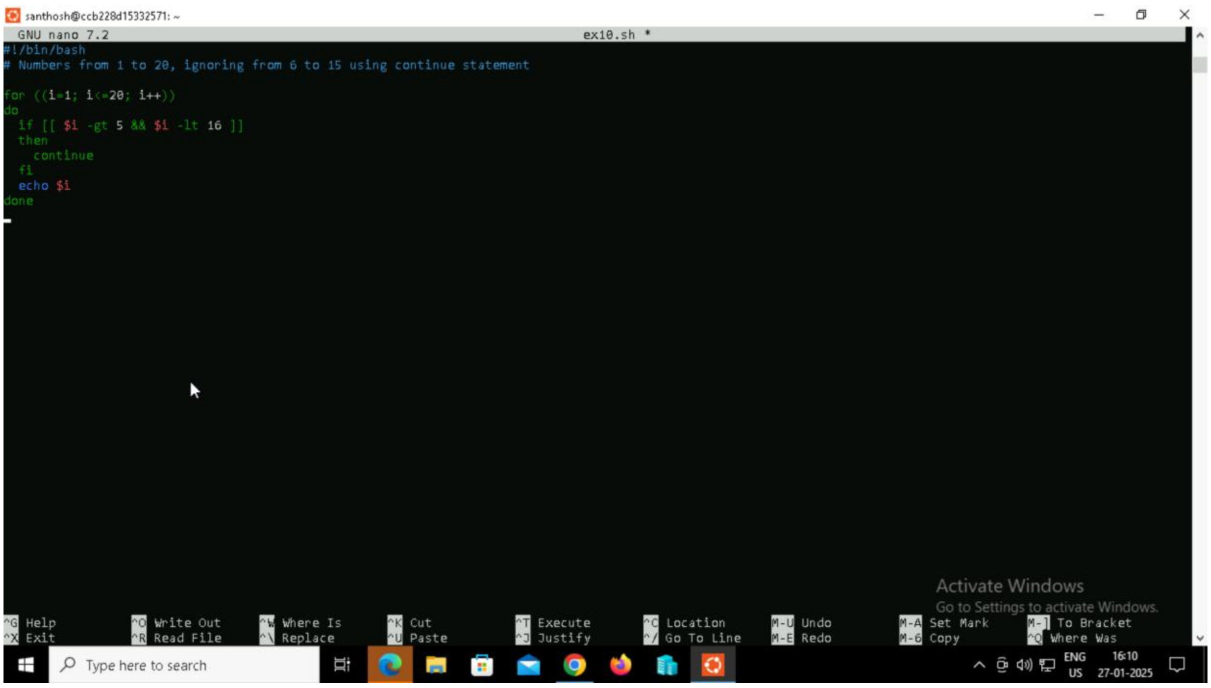
For Loop with a Continue Statement

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



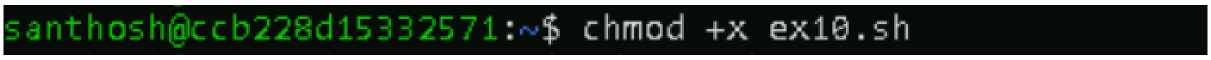
```
santhosh@ccb228d15332571:~$ touch ex10.sh
santhosh@ccb228d15332571:~$ nano ex10.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



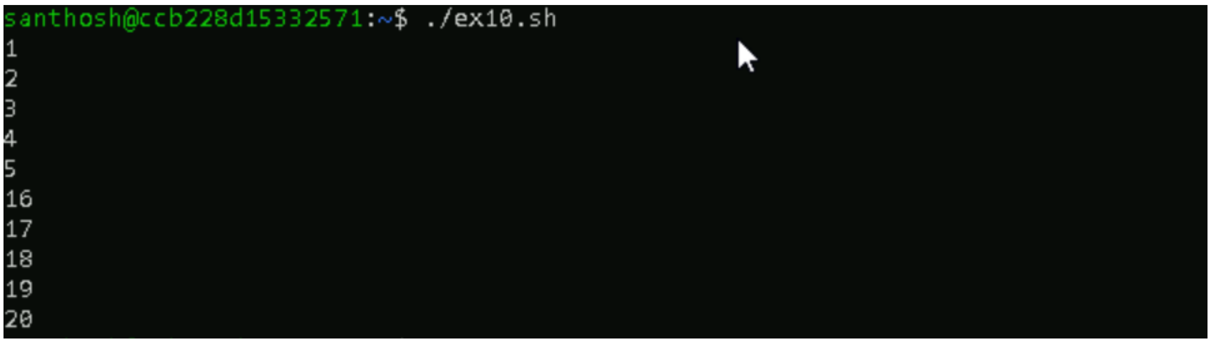
```
#!/bin/bash
# Numbers from 1 to 20, ignoring from 6 to 15 using continue statement

for ((i=1; i<=20; i++))
do
  if [[ $i -gt 5 && $i -lt 16 ]]
  then
    continue
  fi
  echo $i
done
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x ex10.sh
```

Step 4: Executing the output.

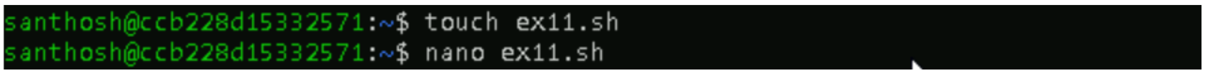


```
santhosh@ccb228d15332571:~$ ./ex10.sh
1
2
3
4
5
16
17
18
19
20
```

Example 11

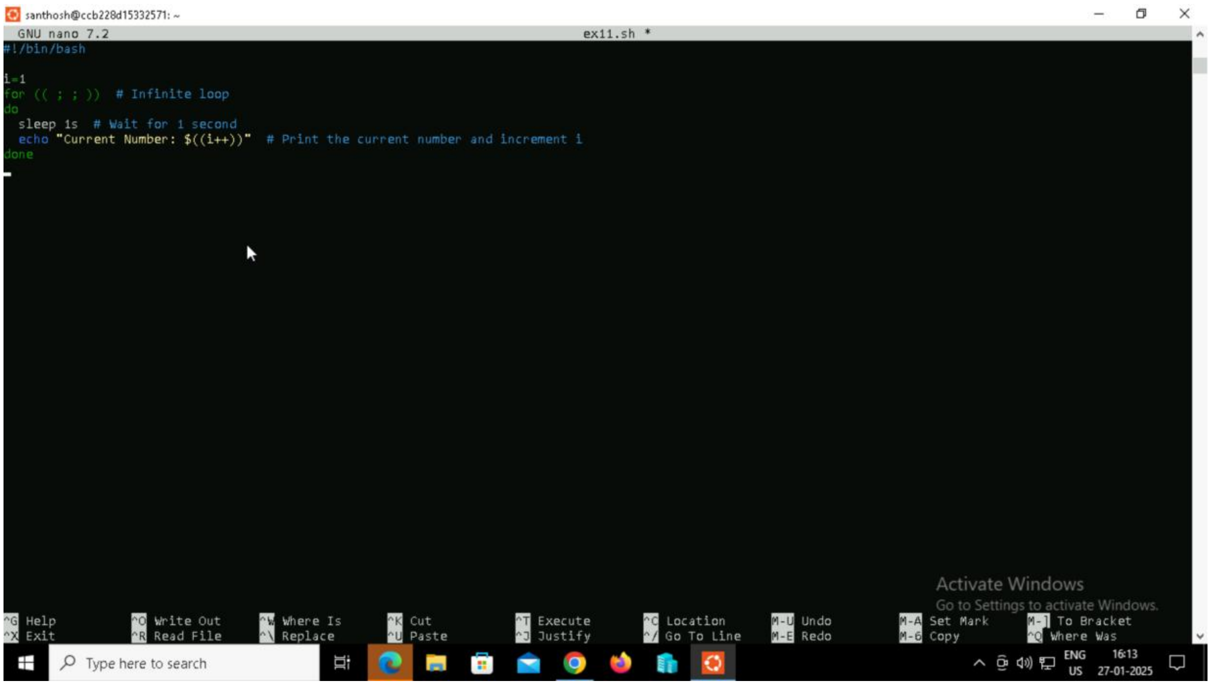
Infinite Bash for Loop

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



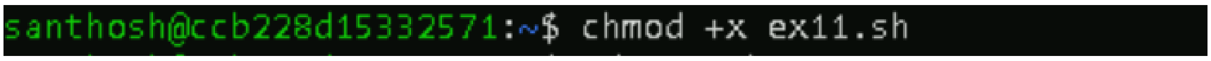
```
santhosh@ccb228d15332571:~$ touch ex11.sh
santhosh@ccb228d15332571:~$ nano ex11.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



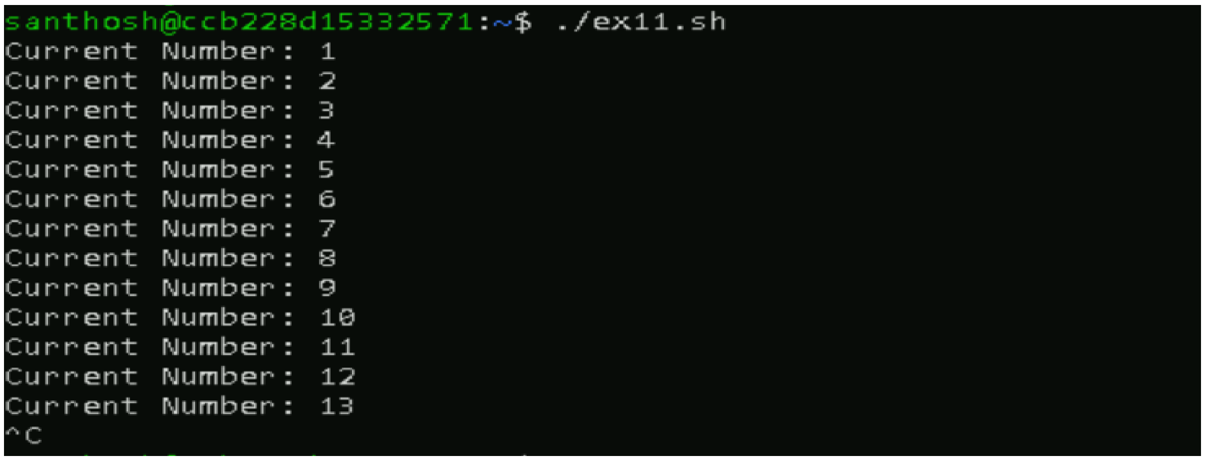
```
  GNU nano 7.2                                    ex11.sh *
#!/bin/bash

i=1
for (( ; ; ))  # Infinite loop
do
  sleep 1s  # Wait for 1 second
  echo "Current Number: $((i++))"  # Print the current number and increment i
done
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x ex11.sh
```

Step 4: Executing the output.



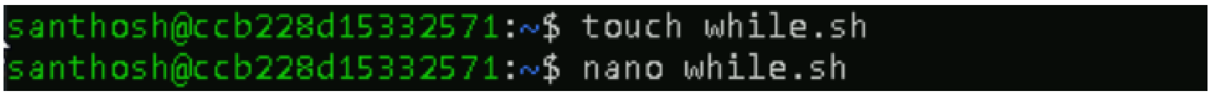
```
santhosh@ccb228d15332571:~$ ./ex11.sh
Current Number: 1
Current Number: 2
Current Number: 3
Current Number: 4
Current Number: 5
Current Number: 6
Current Number: 7
Current Number: 8
Current Number: 9
Current Number: 10
Current Number: 11
Current Number: 12
Current Number: 13
^C
```

BASH While Loop

The bash while loop can be defined as a control flow statement which allows executing the given set of commands repeatedly as long as the applied condition evaluates to true. For example, we can either run echo command many times or just read a text file line by line and process the result by using while loop in Bash.

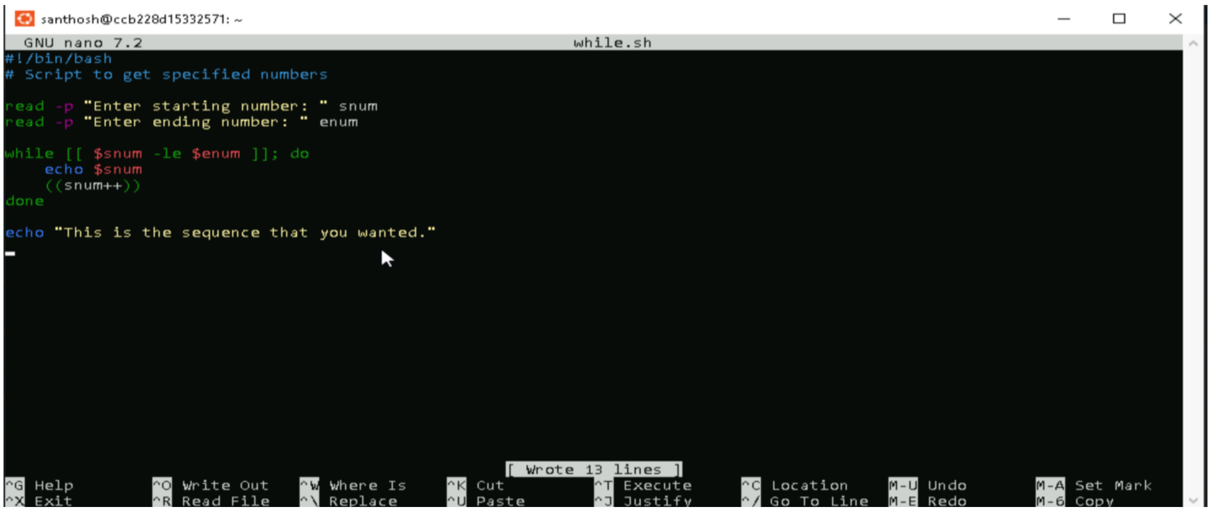
While Loop with Single Condition

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



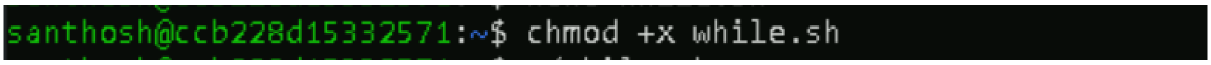
```
santhosh@ccb228d15332571:~$ touch while.sh
santhosh@ccb228d15332571:~$ nano while.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



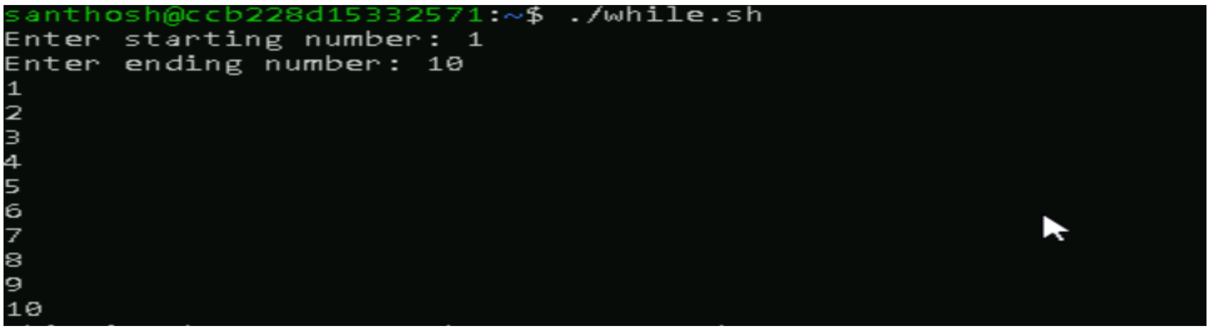
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                          while.sh
#!/bin/bash
# Script to get specified numbers

read -p "Enter starting number: " snum
read -p "Enter ending number: " enum

while [[ $snum -le $enum ]]; do
    echo $snum
    ((snum++))
done

echo "This is the sequence that you wanted."

                              [ Wrote 13 lines ]
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x while.sh
```

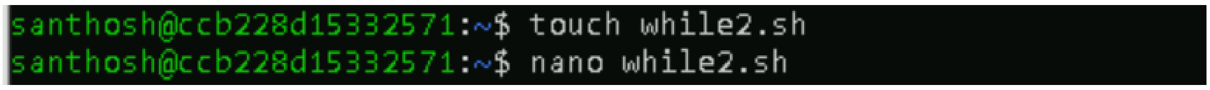
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./while.sh
Enter starting number: 1
Enter ending number: 10
1
2
3
4
5
6
7
8
9
10
```

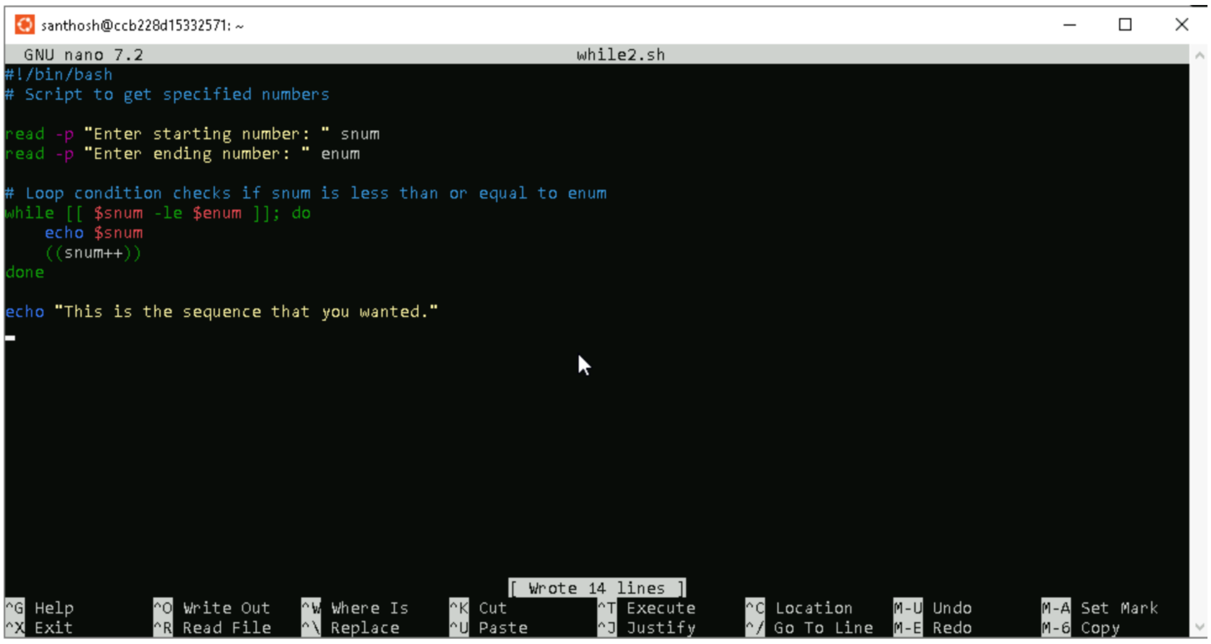
While Loop with Multiple Conditions

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



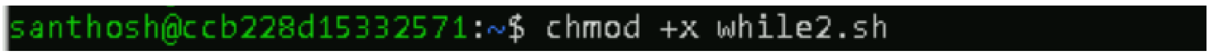
```
santhosh@ccb228d15332571:~$ touch while2.sh
santhosh@ccb228d15332571:~$ nano while2.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



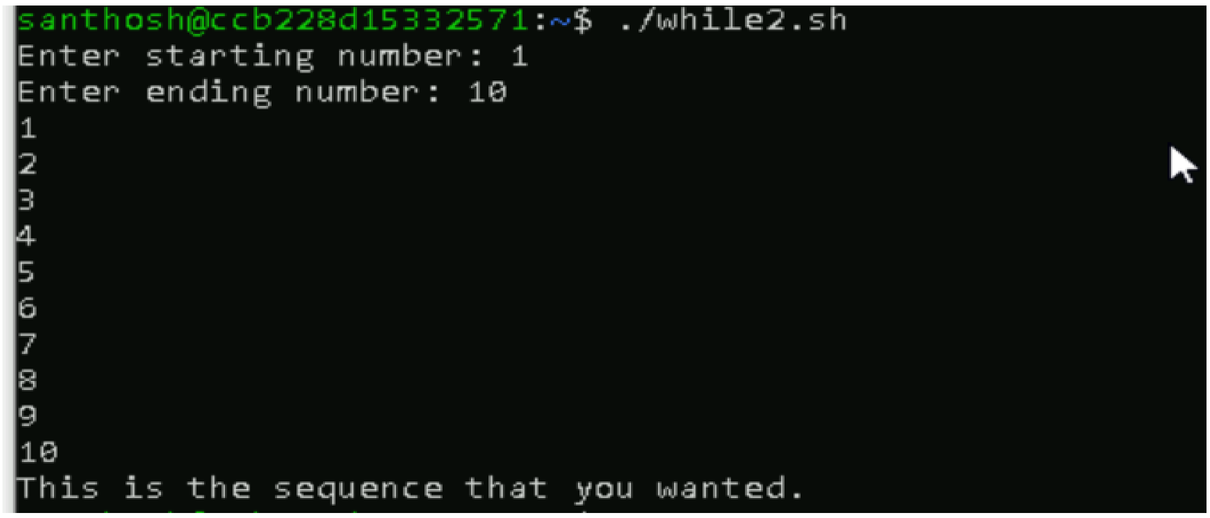
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                                   while2.sh
#!/bin/bash
# Script to get specified numbers

read -p "Enter starting number: " snum
read -p "Enter ending number: " enum

# Loop condition checks if snum is less than or equal to enum
while [[ $snum -le $enum ]]; do
    echo $snum
    ((snum++))
done

echo "This is the sequence that you wanted."

                                          [ Wrote 14 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x while2.sh
```

Step 4: Executing the output.

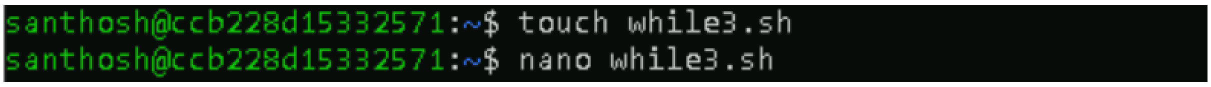


```
santhosh@ccb228d15332571:~$ ./while2.sh
Enter starting number: 1
Enter ending number: 10
1
2
3
4
5
6
7
8
9
10
This is the sequence that you wanted.
```

Infinite While Loop

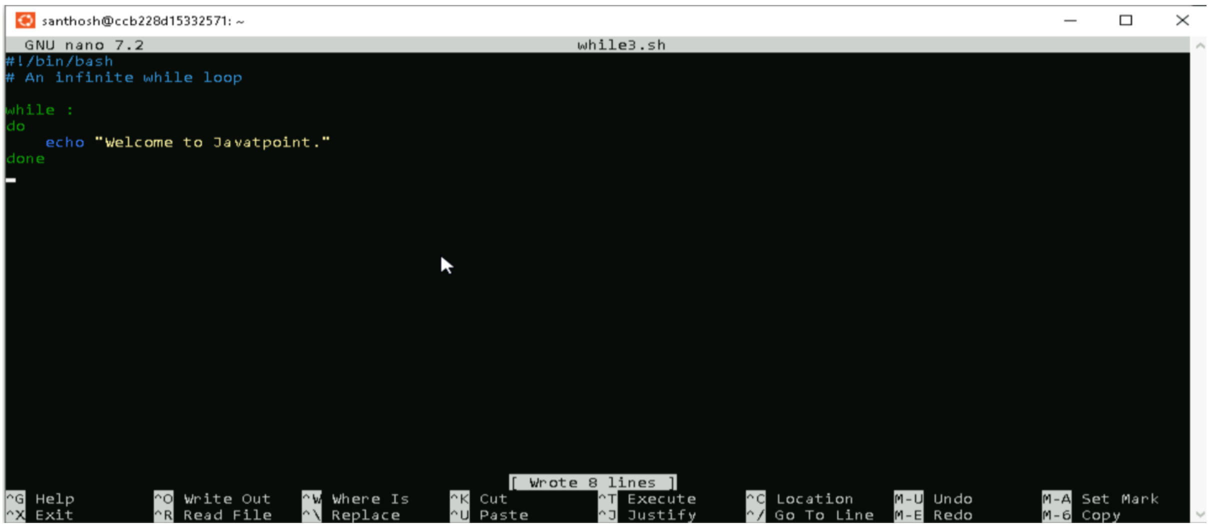
An infinite loop is a loop that has no ending or termination. If the condition always evaluates to true, it creates an infinite loop. The loop will execute continuously until it is forcefully stopped using CTRL+C :

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



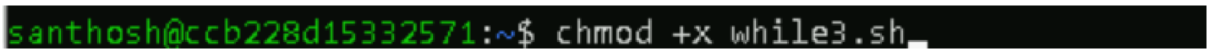
```
santhosh@ccb228d15332571:~$ touch while3.sh
santhosh@ccb228d15332571:~$ nano while3.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



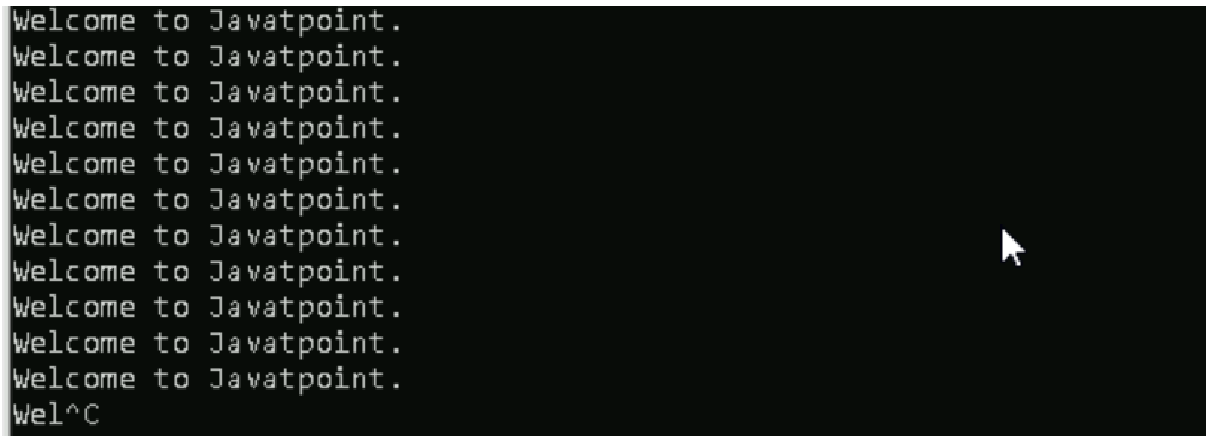
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                          while3.sh
#!/bin/bash
# An infinite while loop

while :
do
    echo "Welcome to Javatpoint."
done

                              [ Wrote 8 lines ]
^G Help     ^O Write Out  ^W Where Is  ^K Cut    ^T Execute  ^C Location    M-U Undo  M-A Set Mark
^X Exit     ^R Read File  ^\ Replace   ^U Paste  ^J Justify  ^/ Go To Line  M-E Redo  M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x while3.sh
```

Step 4: Executing the output.

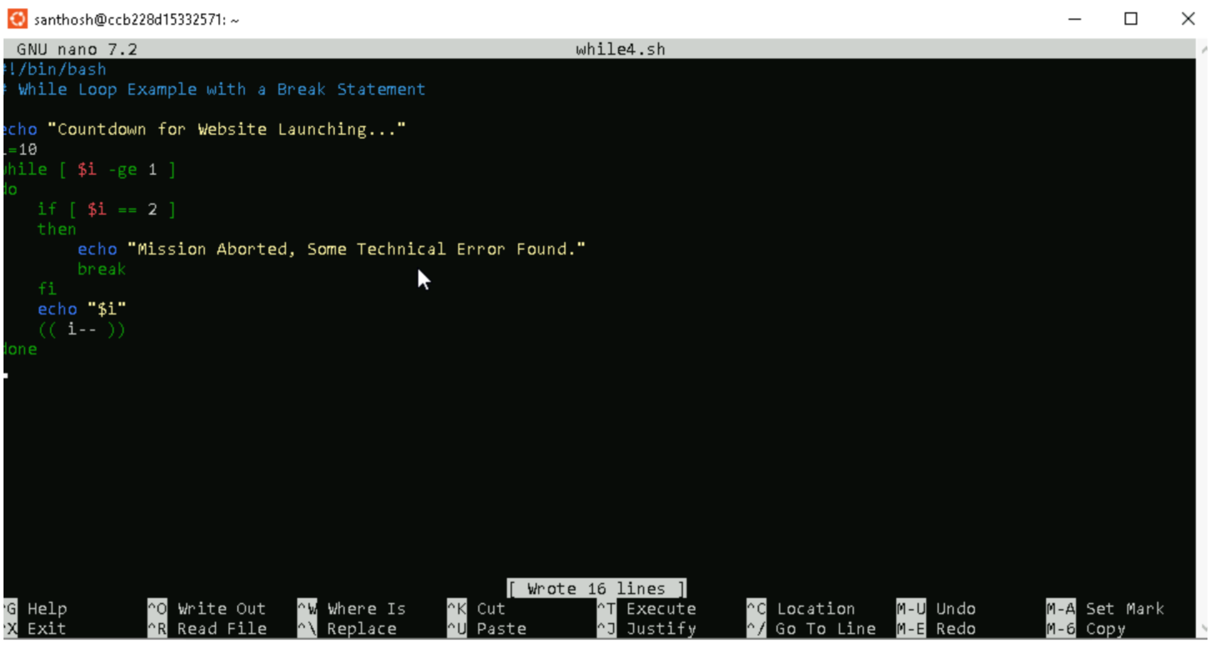


```
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Welcome to Javatpoint.
Wel^C
```

While Loop with a Break Statement

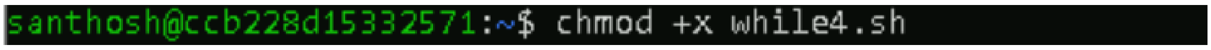
Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



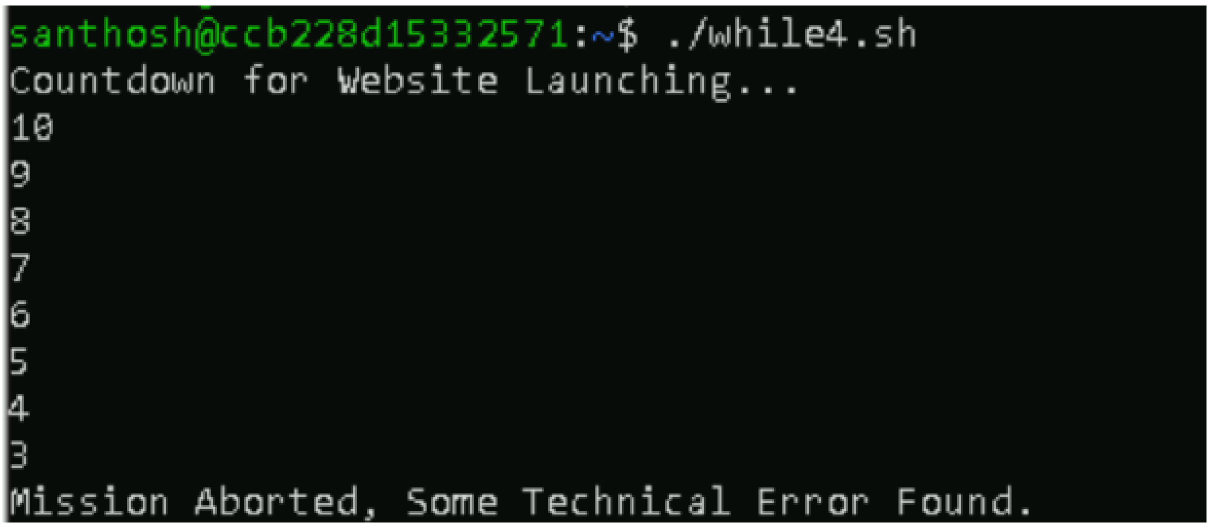
```
santhosh@ccb228d15332571: ~
 GNU nano 7.2                                         while4.sh
!/bin/bash
 While Loop Example with a Break Statement

cho "Countdown for Website Launching..."
=10
hile [ $i -ge 1 ]
o
   if [ $i == 2 ]
   then
       echo "Mission Aborted, Some Technical Error Found."
       break
   fi
   echo "$i"
   (( i-- ))
one

                              [ Wrote 16 lines ]
G Help       ^O Write Out  ^W Where Is   ^K Cut       ^T Execute   ^C Location   M-U Undo     M-A Set Mark
X Exit       ^R Read File  ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line M-E Redo     M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x while4.sh
```

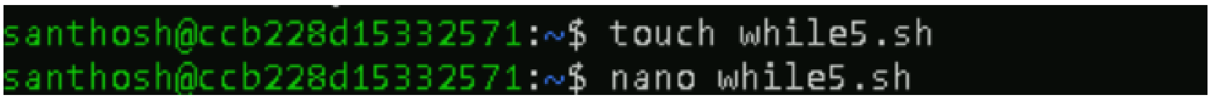
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./while4.sh
Countdown for Website Launching...
10
9
8
7
6
5
4
3
Mission Aborted, Some Technical Error Found.
```

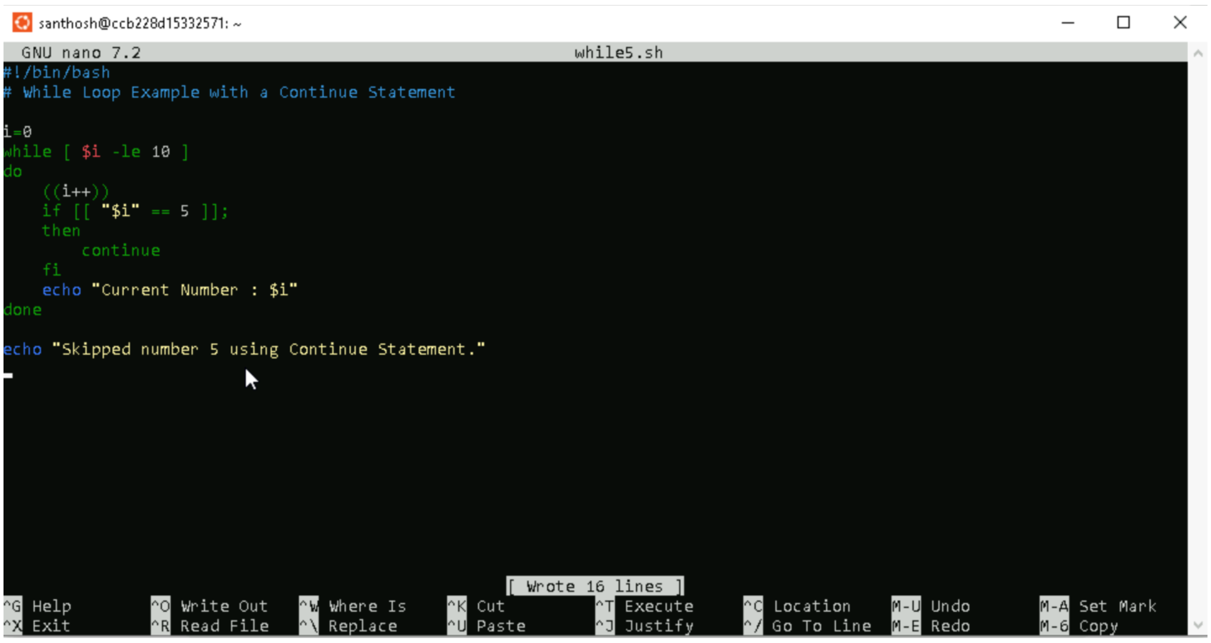
While Loop with a Continue Statement

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



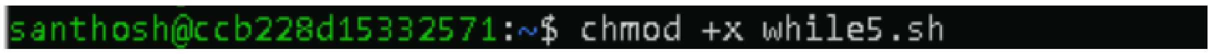
```
santhosh@ccb228d15332571:~$ touch while5.sh
santhosh@ccb228d15332571:~$ nano while5.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



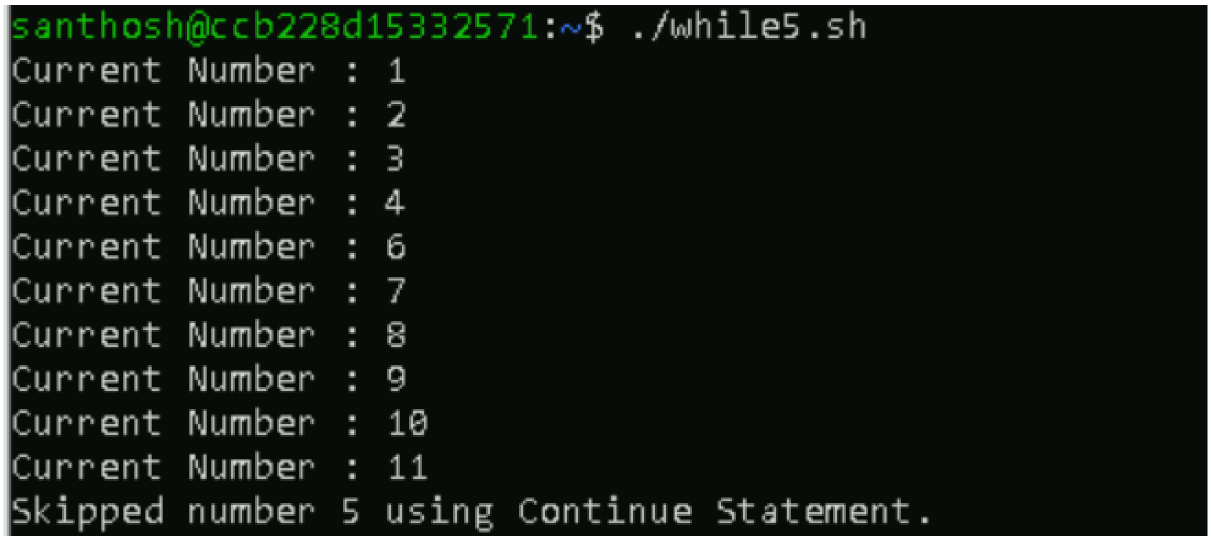
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                          while5.sh
#!/bin/bash
# While Loop Example with a Continue Statement

i=0
while [ $i -le 10 ]
do
    ((i++))
    if [[ "$i" == 5 ]];
    then
        continue
    fi
    echo "Current Number : $i"
done

echo "Skipped number 5 using Continue Statement."

                              [ Wrote 16 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute   ^C Location   M-U Undo     M-A Set Mark
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line M-E Redo     M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x while5.sh
```

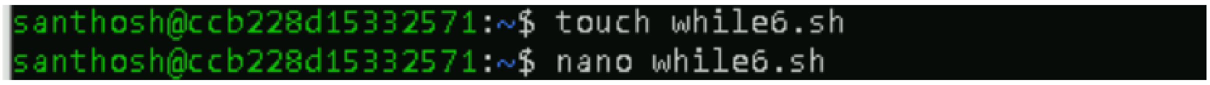
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./while5.sh
Current Number : 1
Current Number : 2
Current Number : 3
Current Number : 4
Current Number : 6
Current Number : 7
Current Number : 8
Current Number : 9
Current Number : 10
Current Number : 11
Skipped number 5 using Continue Statement.
```

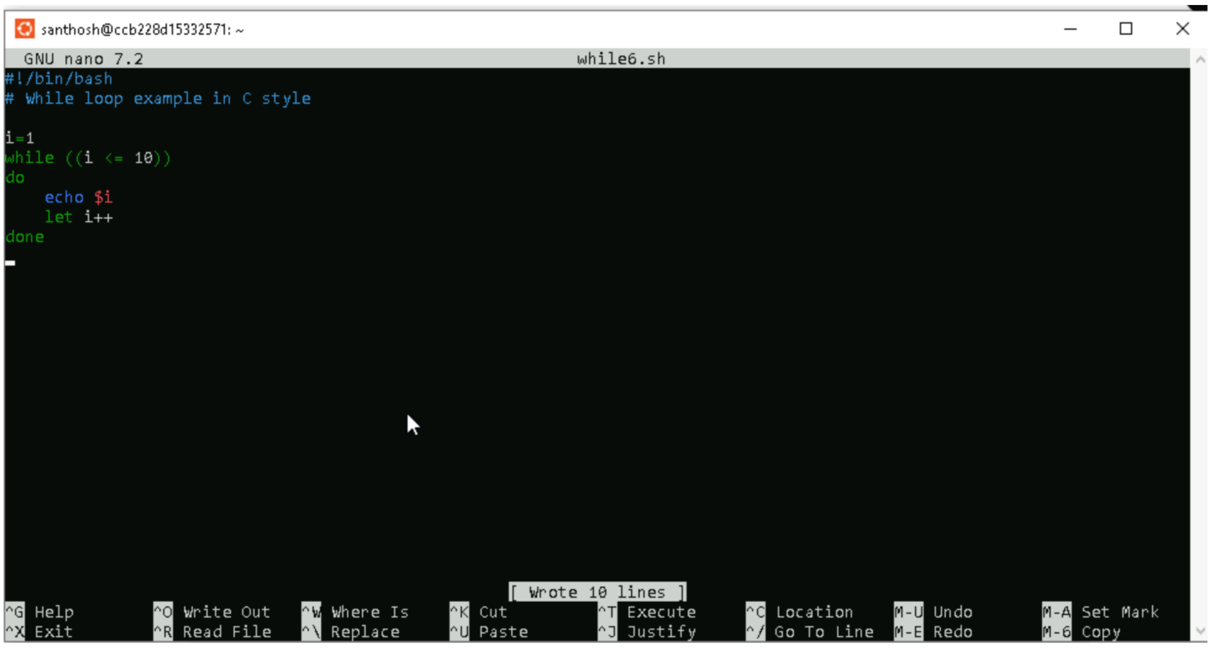
While Loop with C-Style

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



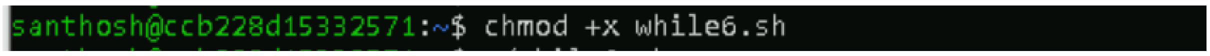
```
santhosh@ccb228d15332571:~$ touch while6.sh
santhosh@ccb228d15332571:~$ nano while6.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



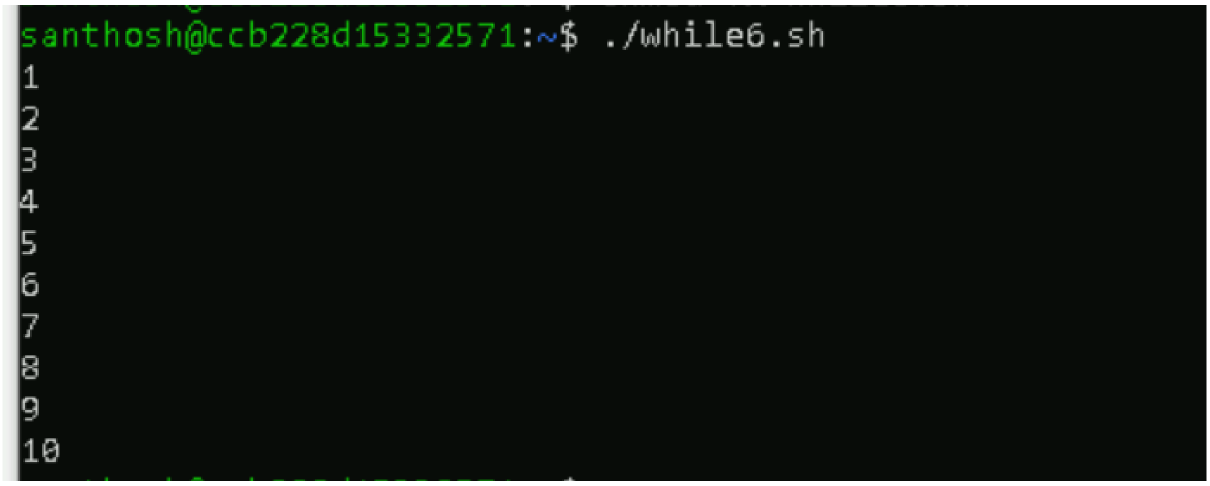
```
santhosh@ccb228d15332571: ~
 GNU nano 7.2                          while6.sh
#!/bin/bash
# While loop example in C style

i=1
while ((i <= 10))
do
    echo $i
    let i++
done

                              [ Wrote 10 lines ]
^G Help      ^O Write Out  ^W Where Is  ^K Cut    ^T Execute  ^C Location    M-U Undo   M-A Set Mark
^X Exit      ^R Read File  ^\ Replace   ^U Paste  ^J Justify  ^/ Go To Line  M-E Redo   M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x while6.sh
```

Step 4: Executing the output.



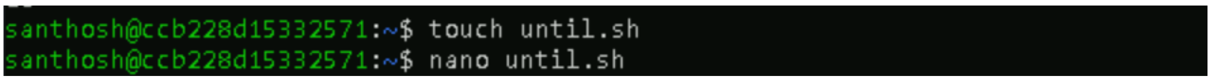
```
santhosh@ccb228d15332571:~$ ./while6.sh
1
2
3
4
5
6
7
8
9
10
```

Bash Until Loop

Bash Until Loop in a bash scripting is used to execute a set of commands repeatedly based on the boolean result of an expression. The set of commands are executed only until the expression evaluates to true. It means that when the expression evaluates to false, a set of commands are executed iteratively. The loop is terminated as soon as the expression evaluates to true for the first time.

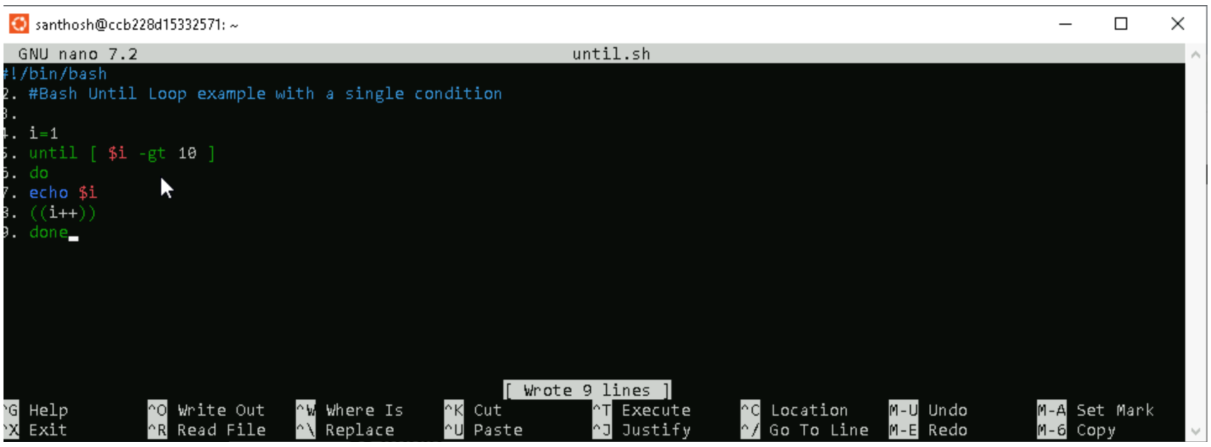
Until Loop with Single Condition

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



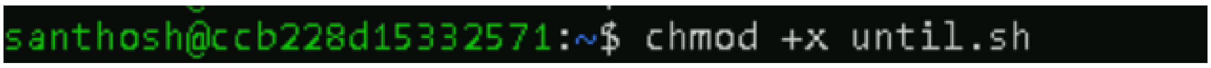
```
santhosh@ccb228d15332571:~$ touch until.sh
santhosh@ccb228d15332571:~$ nano until.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



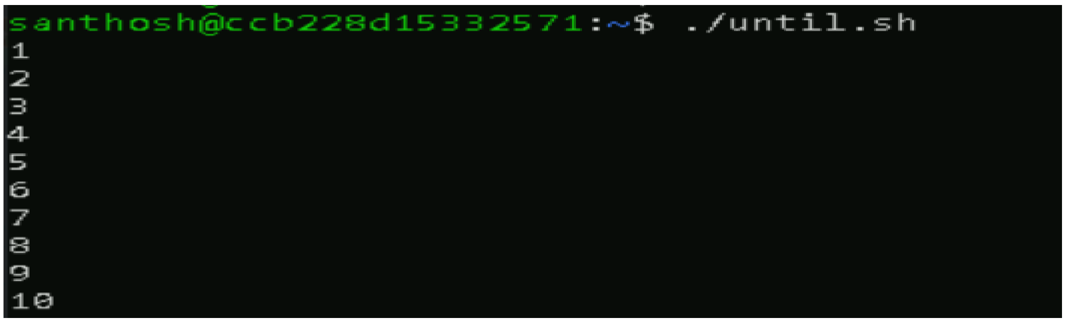
```
  GNU nano 7.2                          until.sh
#!/bin/bash
#Bash Until Loop example with a single condition

i=1
until [ $i -gt 10 ]
do
echo $i
((i++))
done
                                   [ Wrote 9 lines ]
^G Help      ^O Write Out  ^W Where Is  ^K Cut      ^T Execute  ^C Location   M-U Undo   M-A Set Mark
^X Exit      ^R Read File  ^\ Replace   ^U Paste    ^J Justify  ^/ Go To Line M-E Redo   M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x until.sh
```

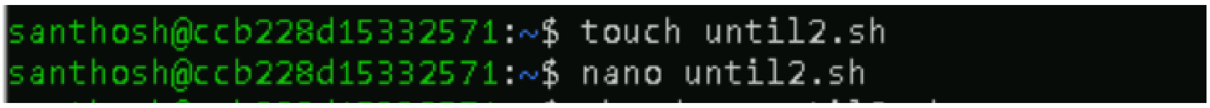
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./until.sh
1
2
3
4
5
6
7
8
9
10
```

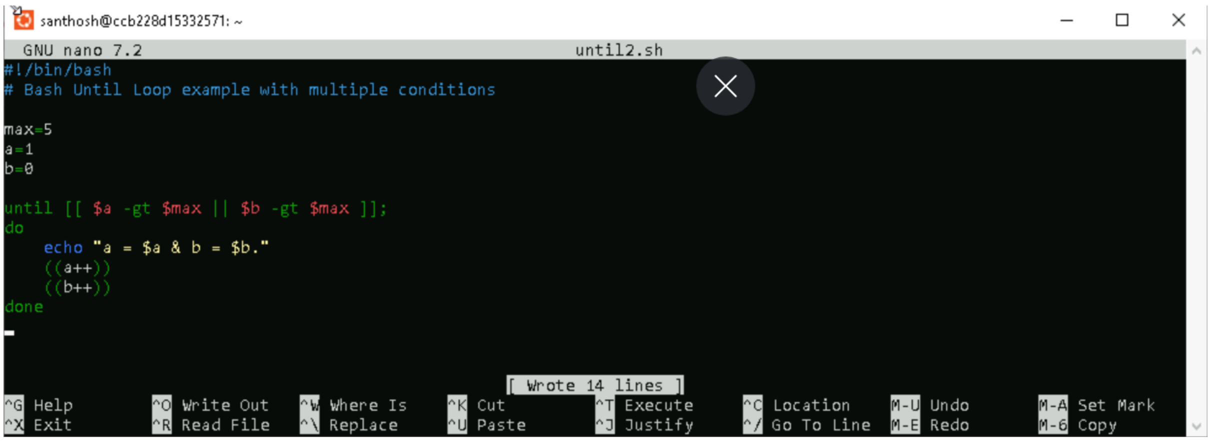
Until Loop with Multiple Conditions

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



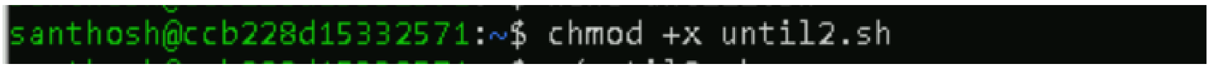
```
santhosh@ccb228d15332571:~$ touch until2.sh
santhosh@ccb228d15332571:~$ nano until2.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



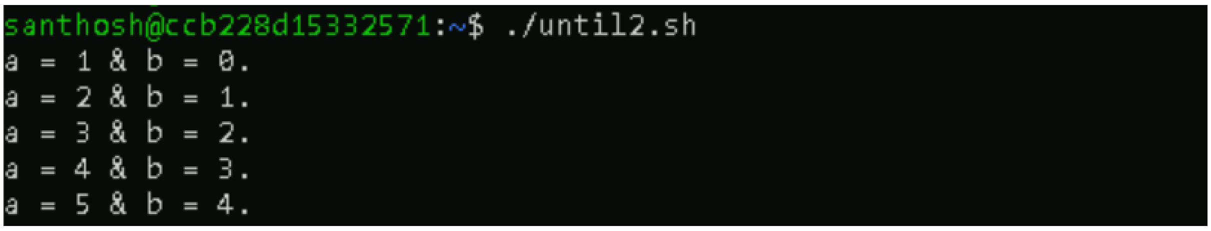
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                          until2.sh
#!/bin/bash
# Bash Until Loop example with multiple conditions

max=5
a=1
b=0

until [[ $a -gt $max || $b -gt $max ]];
do
    echo "a = $a & b = $b."
    ((a++))
    ((b++))
done

                                  [ Wrote 14 lines ]
^G Help      ^O Write Out  ^W Where Is  ^K Cut    ^T Execute  ^C Location   M-U Undo  M-A Set Mark
^X Exit      ^R Read File  ^\ Replace   ^U Paste  ^J Justify  ^/ Go To Line M-E Redo  M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x until2.sh
```

Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./until2.sh
a = 1 & b = 0.
a = 2 & b = 1.
a = 3 & b = 2.
a = 4 & b = 3.
a = 5 & b = 4.
```

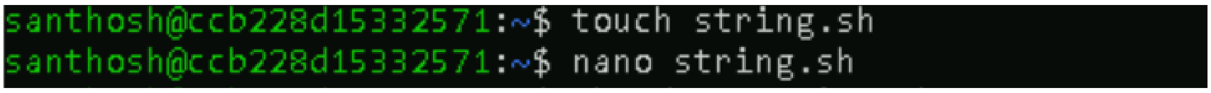
Bash String

Bash String is a data type such as as an integer or floating-point unit. It is used to represent text rather than numbers. It is a combination of a set of characters that may also contain numbers.

Equal Operator

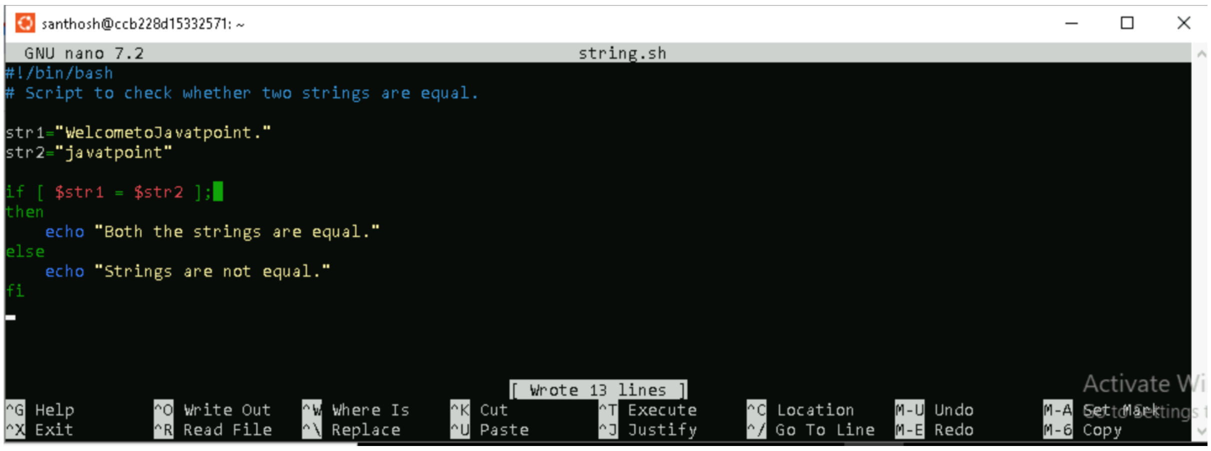
An equal operator (=) is used to check whether two strings are equal.

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



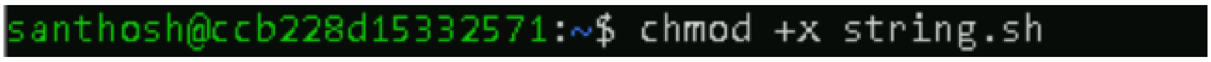
```
santhosh@ccb228d15332571:~$ touch string.sh
santhosh@ccb228d15332571:~$ nano string.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



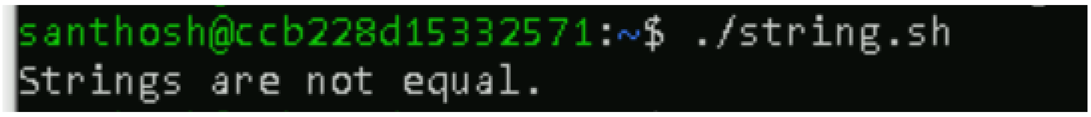
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                          string.sh
#!/bin/bash
# Script to check whether two strings are equal.

str1="WelcometoJavatpoint."
str2="javatpoint"

if [ $str1 = $str2 ];
then
    echo "Both the strings are equal."
else
    echo "Strings are not equal."
fi

                                   [ Wrote 13 lines ]
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x string.sh
```

Step 4: Executing the output.

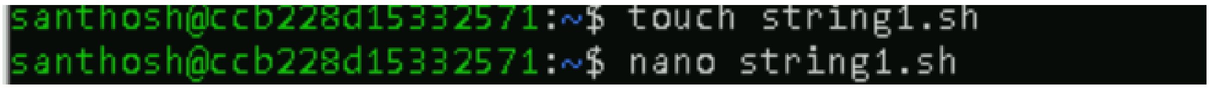


```
santhosh@ccb228d15332571:~$ ./string.sh
Strings are not equal.
```

Not Equal Operator

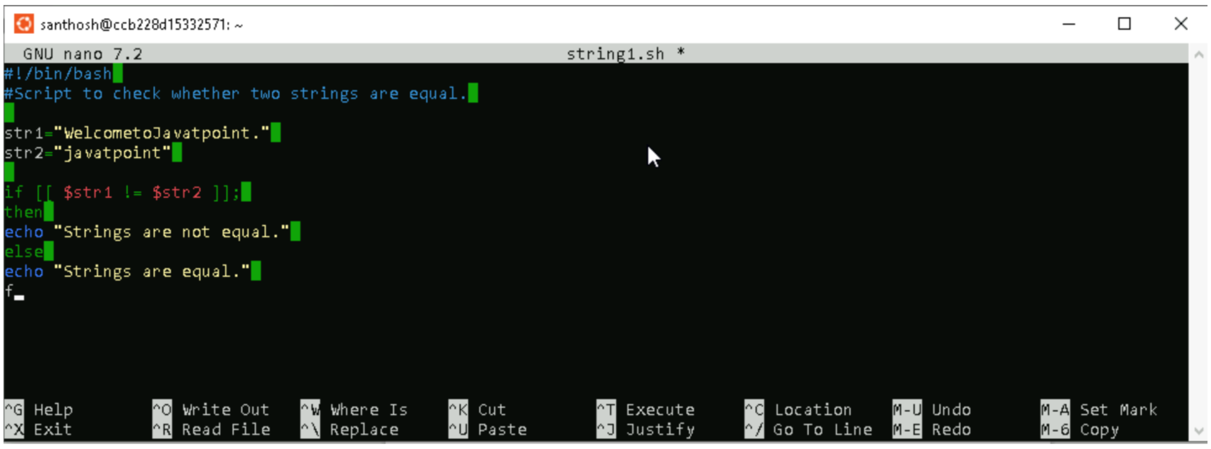
Not equal operator (!=) is used to define that strings are not equal.

Step 1: Creating a bash script using touch command and adding the script bby editing the file using nano command.



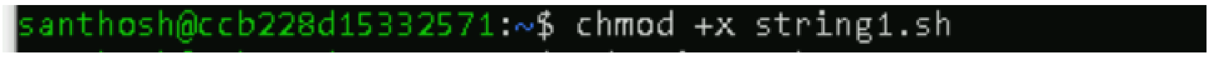
```
santhosh@ccb228d15332571:~$ touch string1.sh
santhosh@ccb228d15332571:~$ nano string1.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



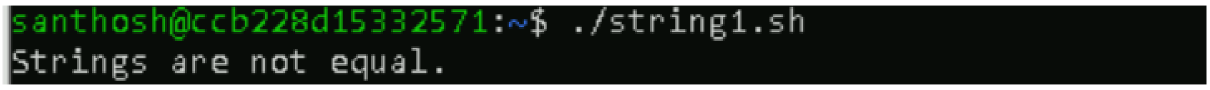
```
santhosh@ccb228d15332571: ~
GNU nano 7.2                     string1.sh *
#!/bin/bash
#Script to check whether two strings are equal.

str1="WelcometoJavatpoint."
str2="javatpoint"

if [[ $str1 != $str2 ]];
then
echo "Strings are not equal."
else
echo "Strings are equal."
fi
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x string1.sh
```

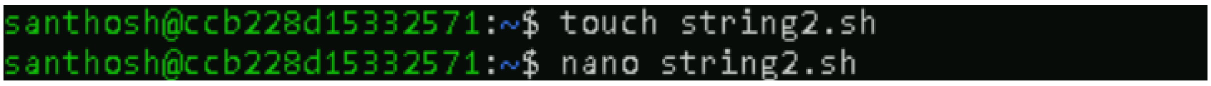
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./string1.sh
Strings are not equal.
```

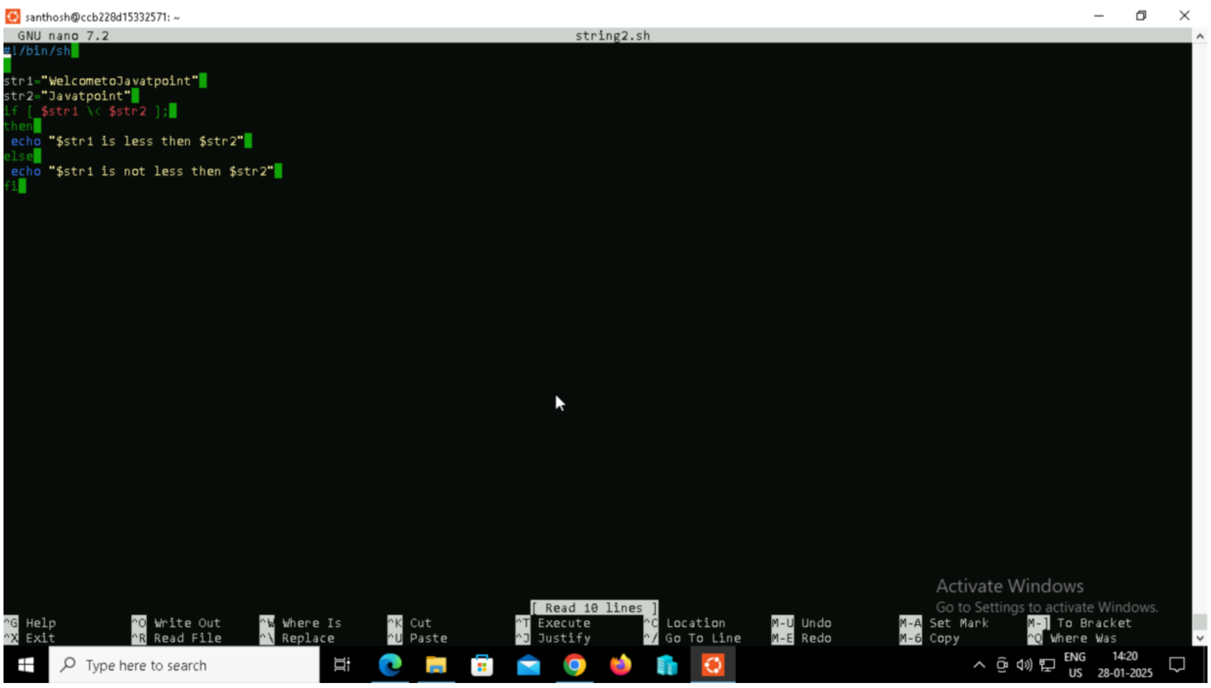
Less than Operator

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



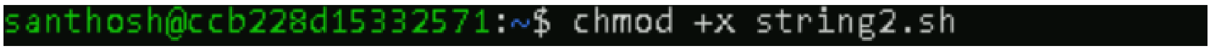
```
santhosh@ccb228d15332571:~$ touch string2.sh
santhosh@ccb228d15332571:~$ nano string2.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



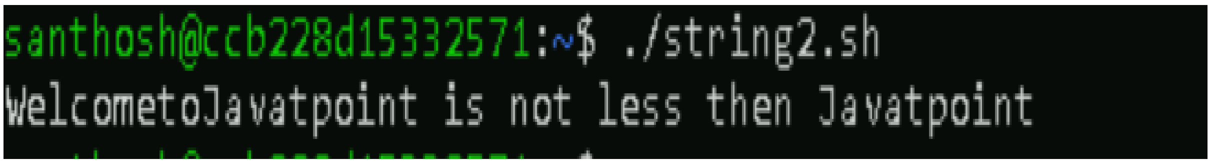
```
GNU nano 7.2                                string2.sh
#!/bin/sh

str1="WelcometoJavatpoint"
str2="Javatpoint"
if [ $str1 \< $str2 ];
then
 echo "$str1 is less then $str2"
else
 echo "$str1 is not less then $str2"
fi

                              [ Read 10 lines ]
^G Help       ^O Write Out   ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo     M-A Set Mark    M-] To Bracket
^X Exit       ^R Read File   ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo     M-6 Copy        ^Q Where Was
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x string2.sh
```

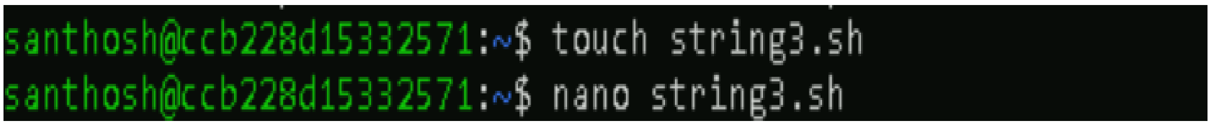
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./string2.sh
WelcometoJavatpoint is not less then Javatpoint
```

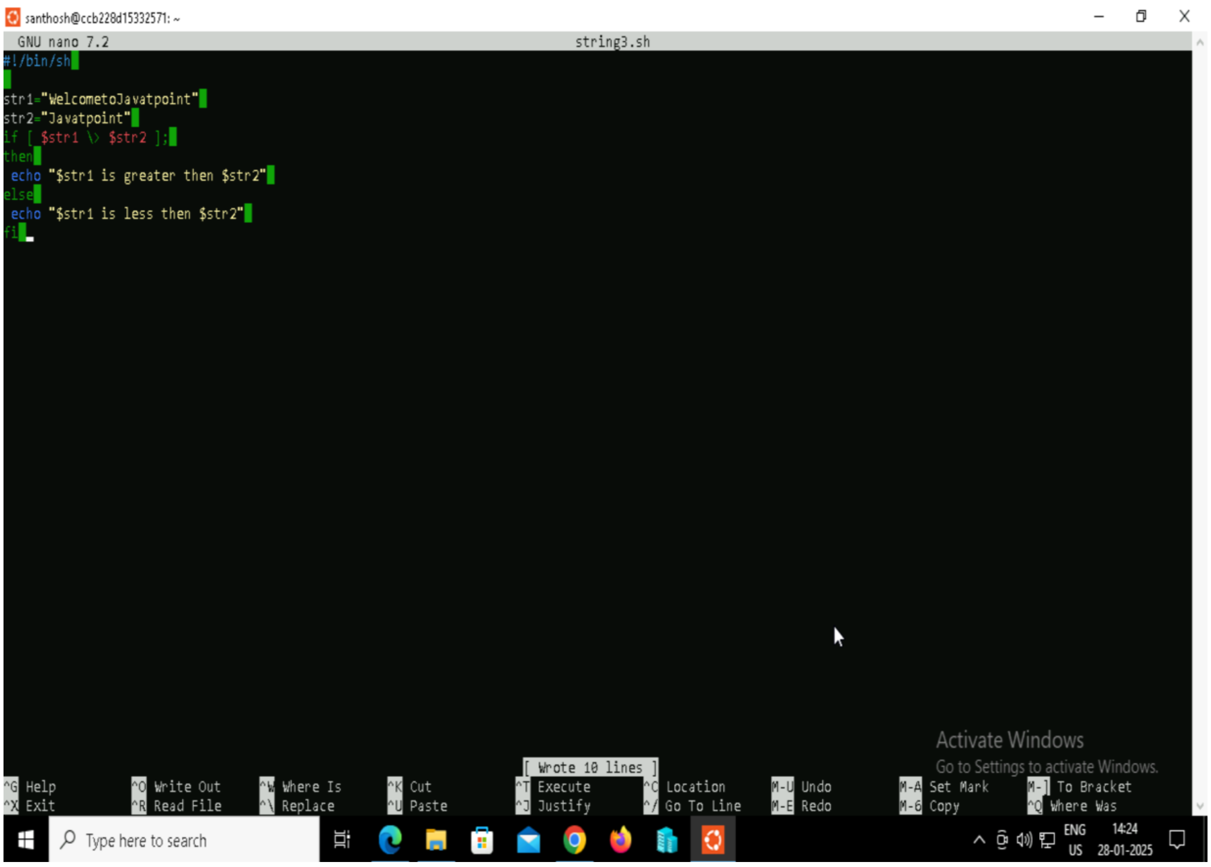
Greater than Operator

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



```
santhosh@ccb228d15332571:~$ touch string3.sh
santhosh@ccb228d15332571:~$ nano string3.sh
```

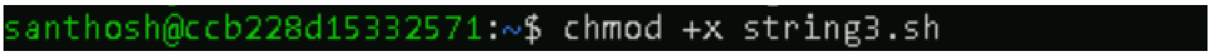
Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



```
GNU nano 7.2                         string3.sh
#!/bin/sh

str1="WelcometoJavatpoint"
str2="Javatpoint"
if [ $str1 \> $str2 ];
then
 echo "$str1 is greater then $str2"
else
 echo "$str1 is less then $str2"
fi
```

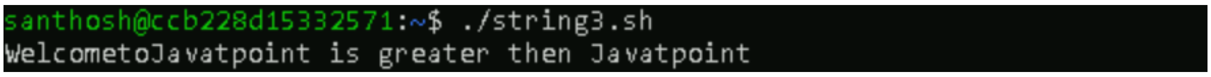
```
[ Wrote 10 lines ]
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x string3.sh
```

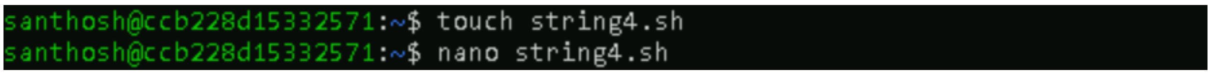
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./string3.sh
WelcometoJavatpoint is greater then Javatpoint
```

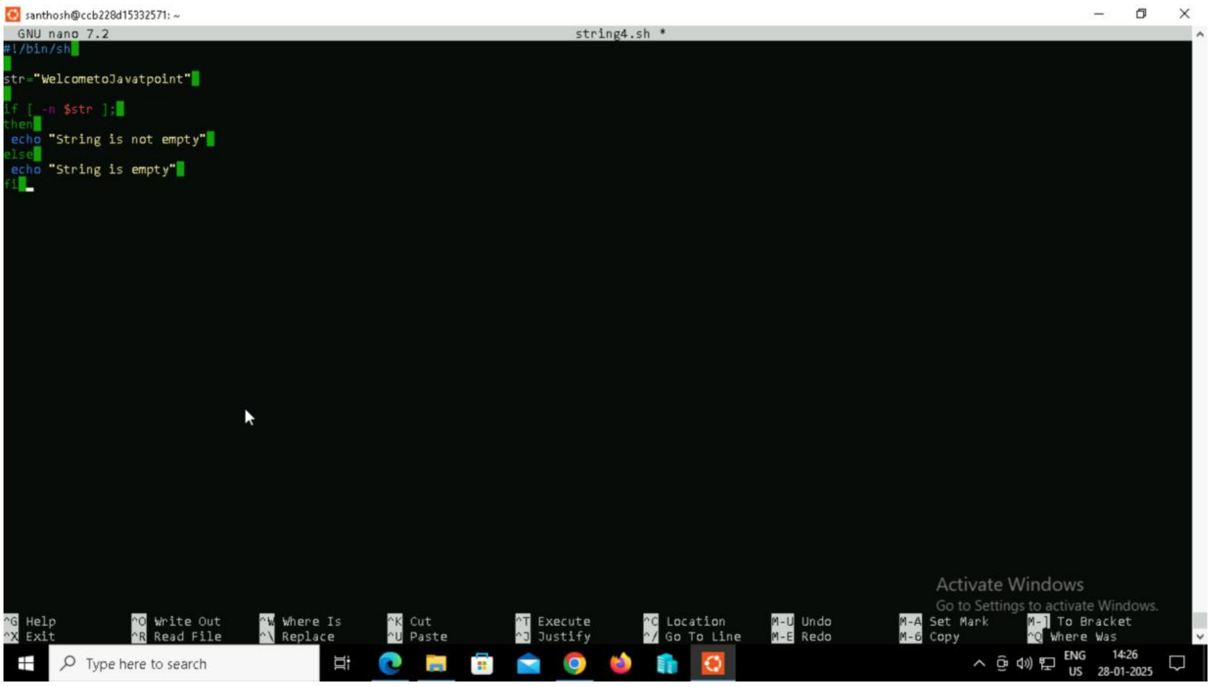
To check if the string length is greater than Zero:

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



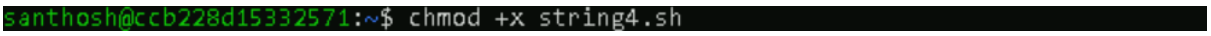
```
santhosh@ccb228d15332571:~$ touch string4.sh
santhosh@ccb228d15332571:~$ nano string4.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



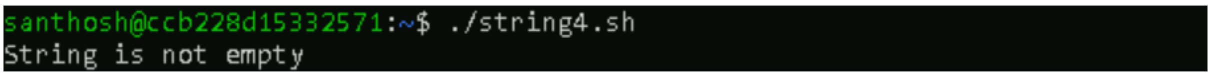
```
#!/bin/sh

str="WelcometoJavatpoint"

if [ -n $str ];
then
 echo "String is not empty"
else
 echo "String is empty"
fi
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x string4.sh
```

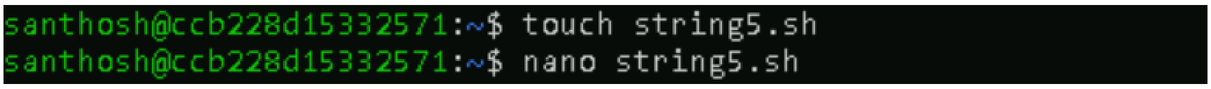
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./string4.sh
String is not empty
```

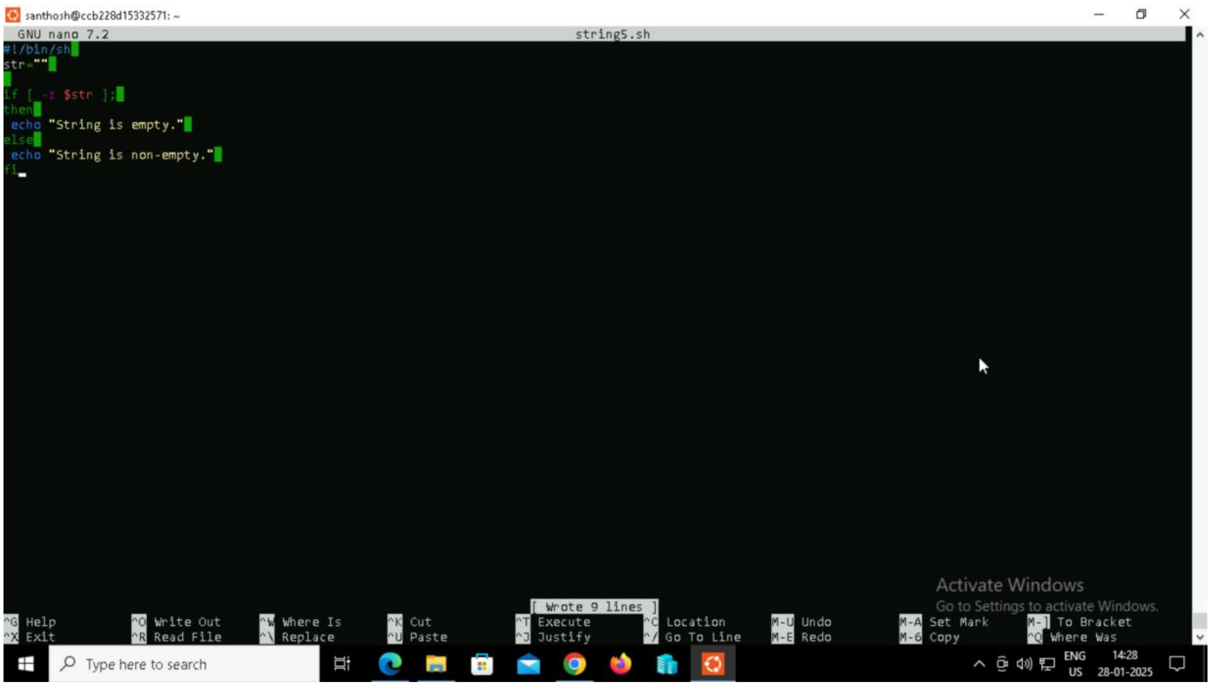
To check if the string length is equal to Zero

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



```
santhosh@ccb228d15332571:~$ touch string5.sh
santhosh@ccb228d15332571:~$ nano string5.sh
```

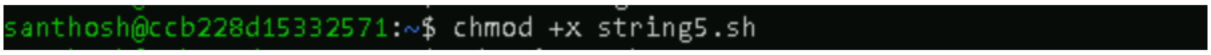
Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



```
GNU nano 7.2                                string5.sh
#!/bin/sh
str=""

if [ -z $str ];
then
 echo "String is empty."
else
 echo "String is non-empty."
fi
```

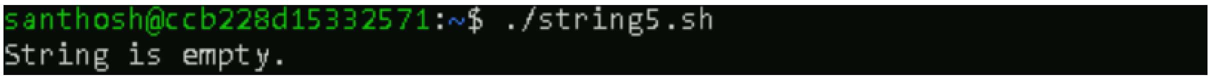
```
[ Wrote 9 lines ]
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x string5.sh
```

Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./string5.sh
String is empty.
```

Bash Find

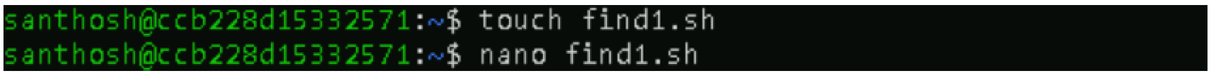
The total number of characters in any string indicates the length of a string. In some cases, we might need to know about the length of a string to perform some specific tasks. Most of the programming languages have their own built-in functions to calculate the number of characters. However, Bash does not contain such type of built-in functions. But there are several ways that we can use to find the length of a string in Bash Scripting.

Bash StringLength

Examples to find String Length in Bash

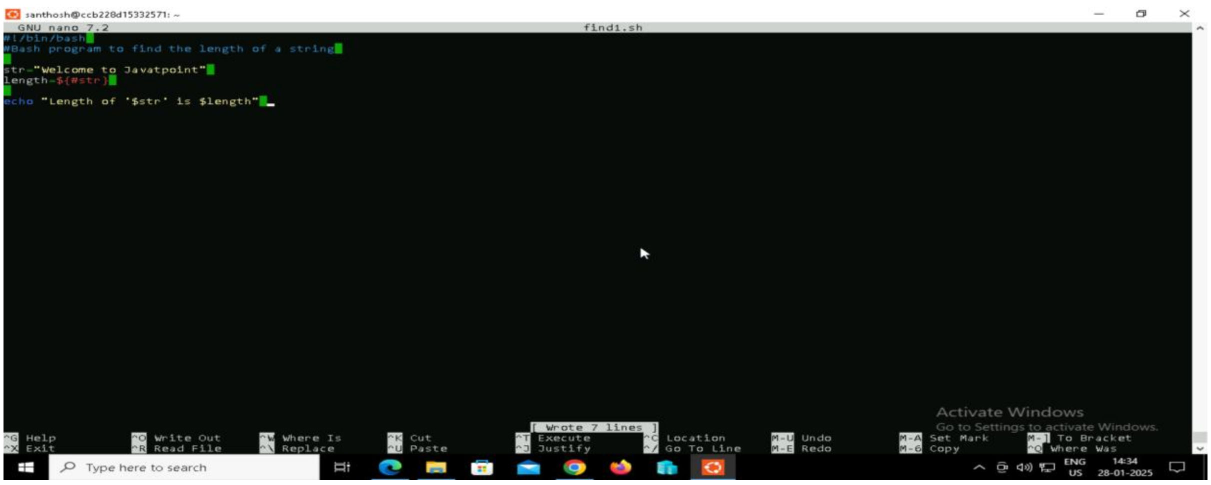
Example 1

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



```
santhosh@ccb228d15332571:~$ touch find1.sh
santhosh@ccb228d15332571:~$ nano find1.sh
```

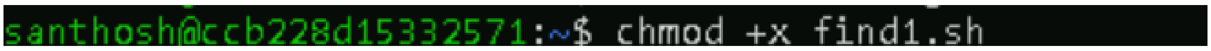
Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



```
#!/bin/bash
#Bash program to find the length of a string

str="Welcome to Javatpoint"
length=${#str}

echo "Length of '$str' is $length"
```

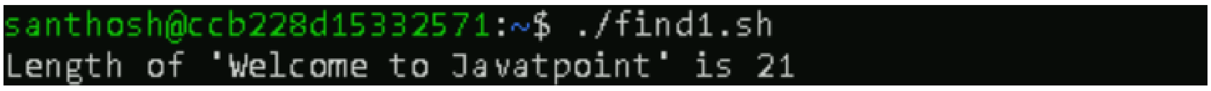
santhosh@ccb228d15332571: ~
GNU nano 7.2
find1.sh
[Wrote 7 lines]
^G Help ^X Exit ^O Write Out ^R Read File ^W Where Is ^\ Replace ^K Cut ^U Paste ^T Execute ^J Justify ^C Location ^/ Go To Line M-U Undo M-E Redo M-A Set Mark M-6 Copy M-] To Bracket ^Q Where Was
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG US 14:34 28-01-2025

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x find1.sh
```

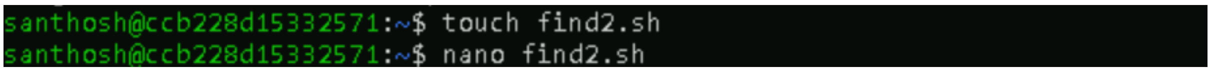
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./find1.sh
Length of 'Welcome to Javatpoint' is 21
```

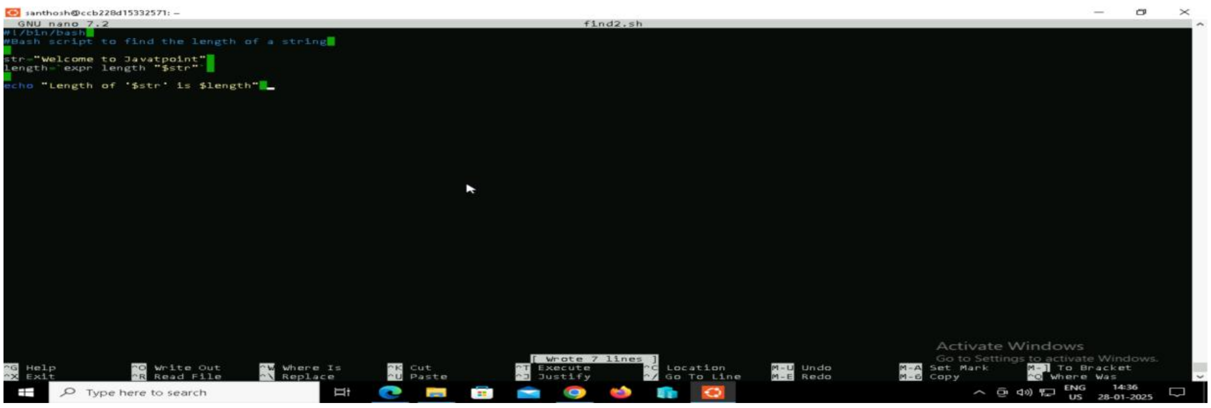
Example 2

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



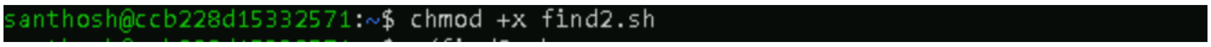
```
santhosh@ccb228d15332571:~$ touch find2.sh
santhosh@ccb228d15332571:~$ nano find2.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



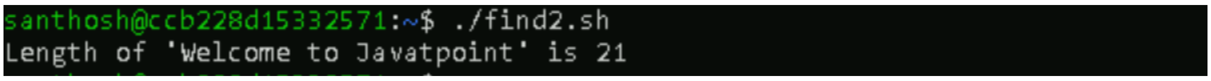
```
#!/bin/bash
#Bash script to find the length of a string

str="Welcome to Javatpoint"
length=`expr length "$str"`

echo "Length of '$str' is $length"
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x find2.sh
```

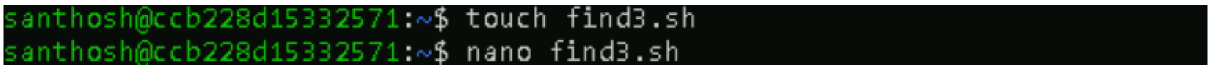
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./find2.sh
Length of 'Welcome to Javatpoint' is 21
```

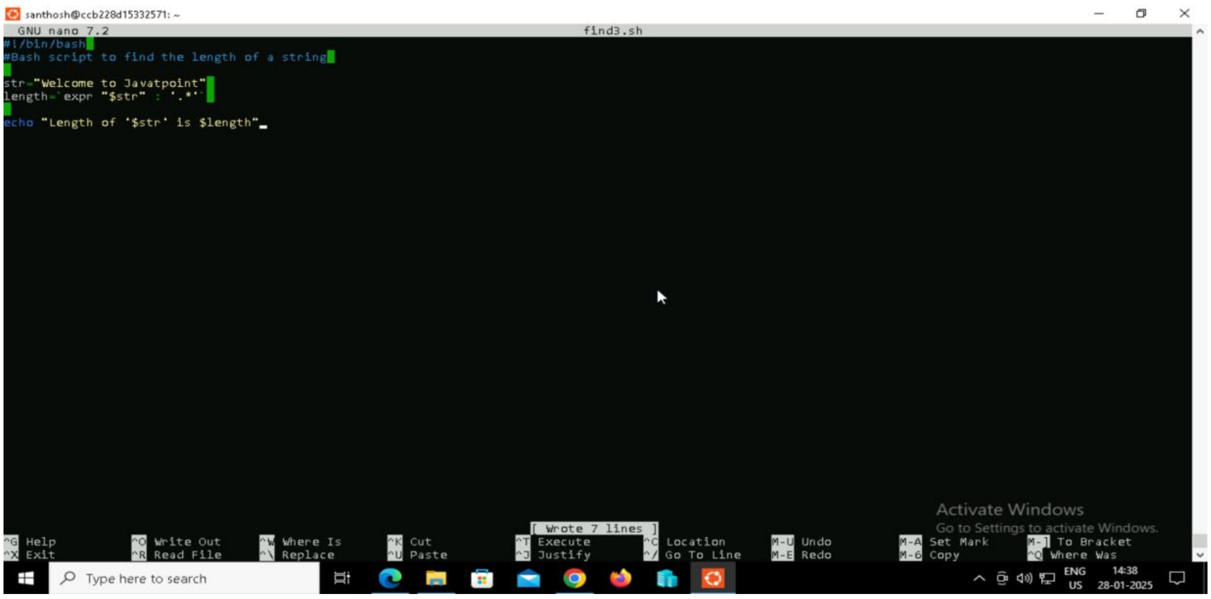
Example 3

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



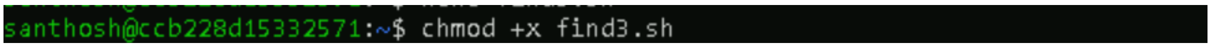
```
santhosh@ccb228d15332571:~$ touch find3.sh
santhosh@ccb228d15332571:~$ nano find3.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



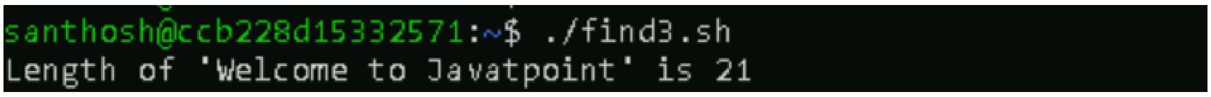
```
  GNU nano 7.2                                   find3.sh
#!/bin/bash
#Bash script to find the length of a string

str="Welcome to Javatpoint"
length=`expr "$str" : '.*'`

echo "Length of '$str' is $length"
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x find3.sh
```

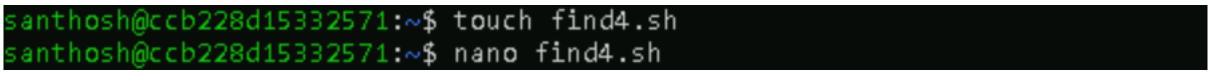
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./find3.sh
Length of 'Welcome to Javatpoint' is 21
```

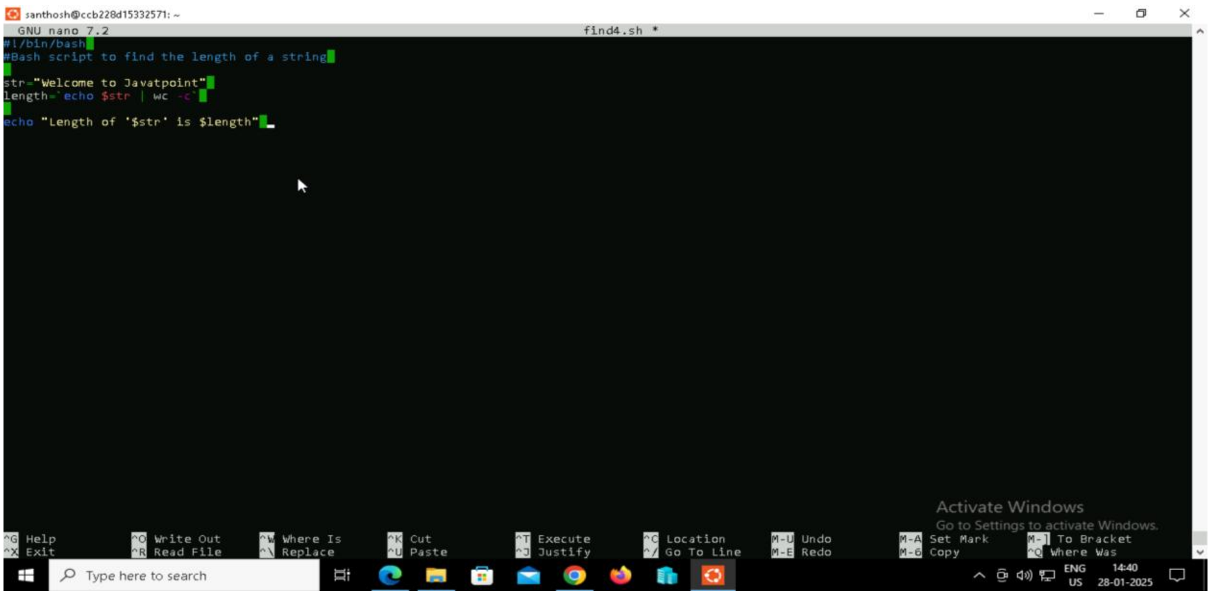
Example 4

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



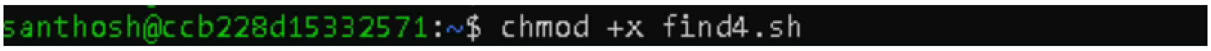
```
santhosh@ccb228d15332571:~$ touch find4.sh
santhosh@ccb228d15332571:~$ nano find4.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



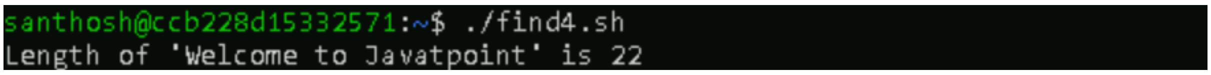
```
  GNU nano 7.2                                   find4.sh *
#!/bin/bash
#Bash script to find the length of a string

str="Welcome to Javatpoint"
length=`echo $str | wc -c`

echo "Length of '$str' is $length"
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x find4.sh
```

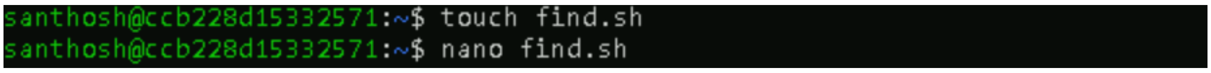
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./find4.sh
Length of 'Welcome to Javatpoint' is 22
```

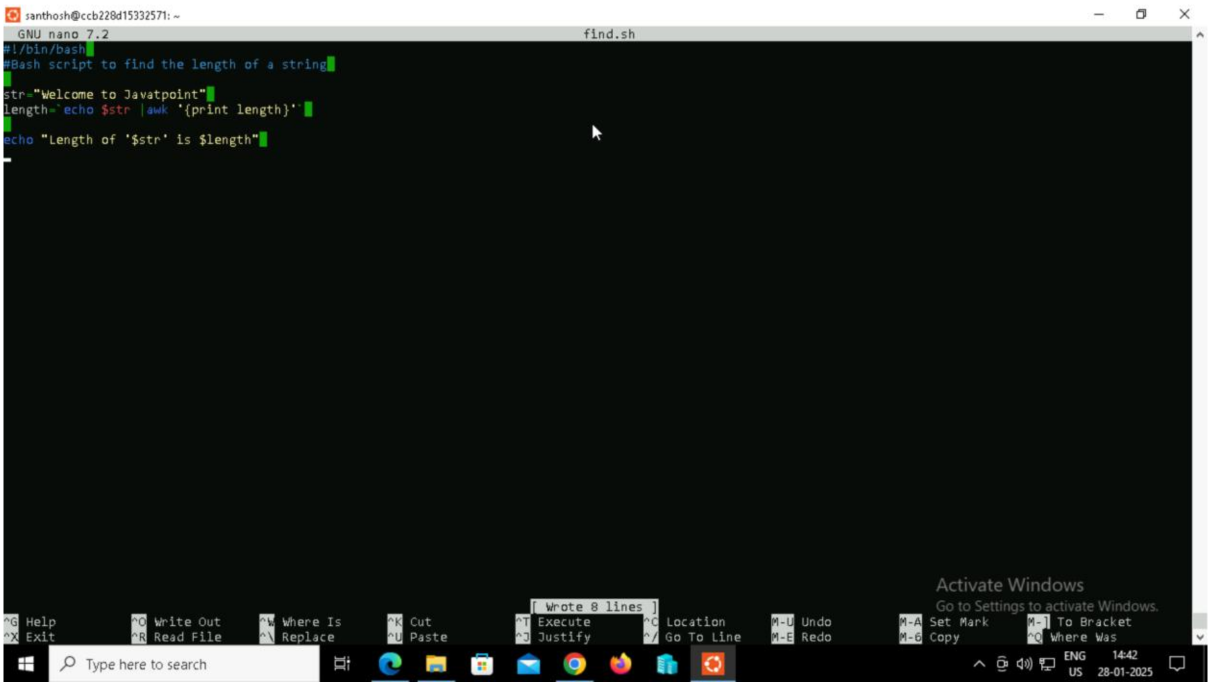
Example 5

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



```
santhosh@ccb228d15332571:~$ touch find.sh
santhosh@ccb228d15332571:~$ nano find.sh
```

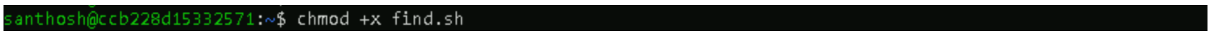
Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



```
  GNU nano 7.2                                   find.sh
#!/bin/bash
#Bash script to find the length of a string

str="Welcome to Javatpoint"
length=`echo $str |awk '{print length}'`

echo "Length of '$str' is $length"

                                        [ Wrote 8 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark  M-] To Bracket
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy      ^Q Where Was
```

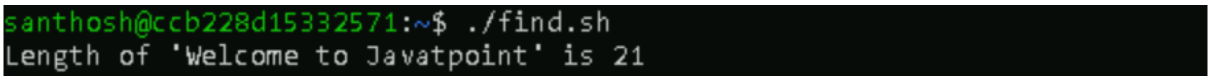
santhosh@ccb228d15332571: ~
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
US
14:42
28-01-2025

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x find.sh
```

Step 4: Executing the output.

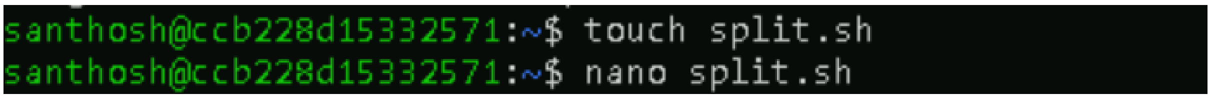


```
santhosh@ccb228d15332571:~$ ./find.sh
Length of 'Welcome to Javatpoint' is 21
```

Bash Split String

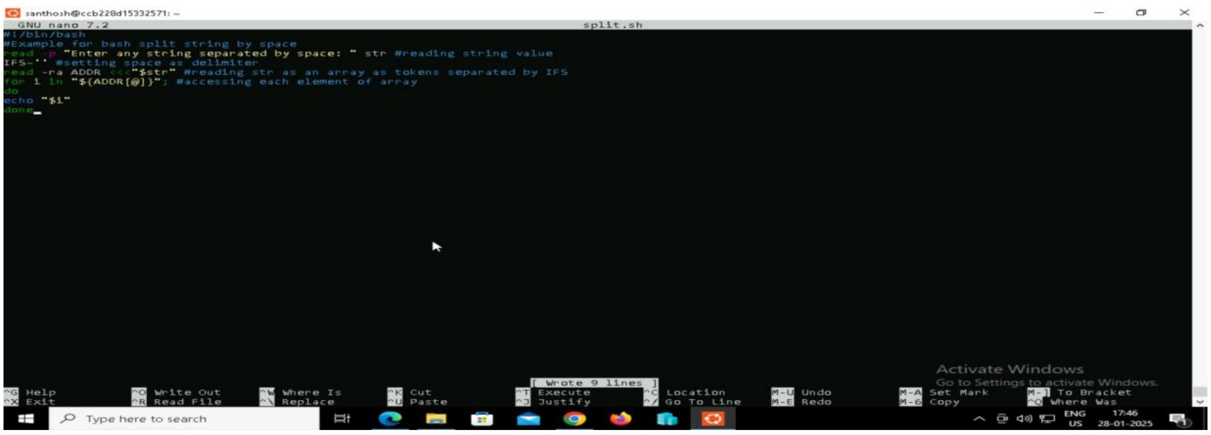
Example 1: Bash Split String by Space

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



```
santhosh@ccb228d15332571:~$ touch split.sh
santhosh@ccb228d15332571:~$ nano split.sh
```

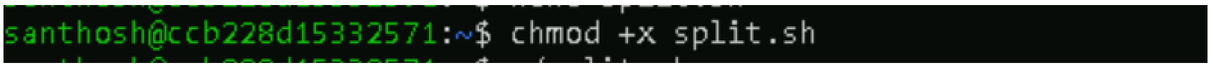
Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



```
#!/bin/bash
#Example for bash split string by space
read -p "Enter any string separated by space: " str #reading string value
IFS=' ' #setting space as delimiter
read -ra ADDR <<<"$str" #reading str as an array as tokens separated by IFS
for i in "${ADDR[@]}"; #accessing each element of array
do
echo "$i"
done
```

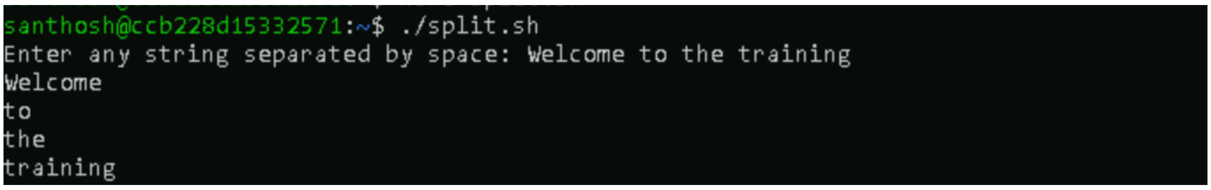
```
[ Wrote 9 lines ]
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x split.sh
```

Step 4: Executing the output.

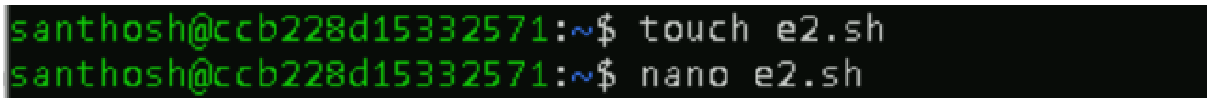


```
santhosh@ccb228d15332571:~$ ./split.sh
Enter any string separated by space: Welcome to the training
Welcome
to
the
training
```

Example 2: Bash Split String by Symbol

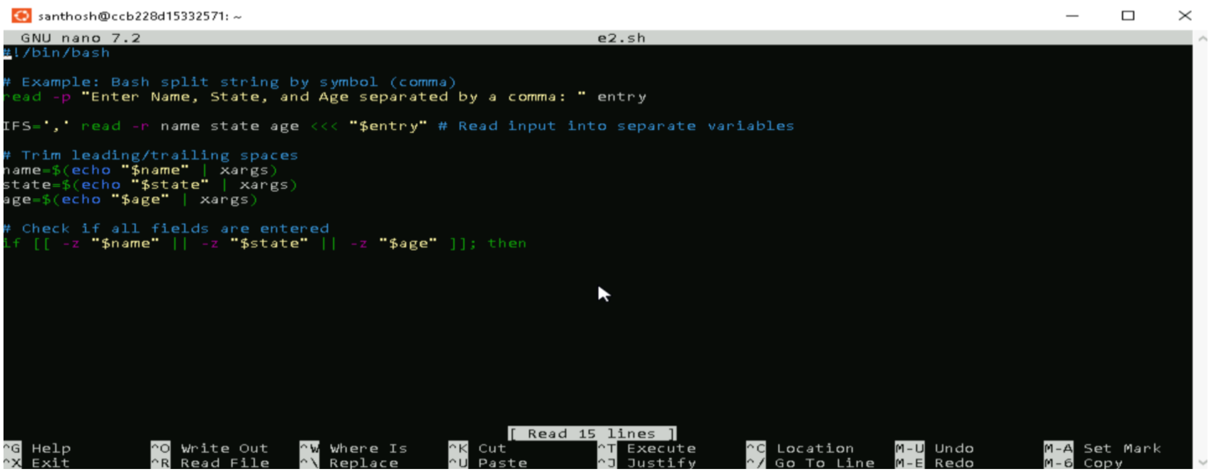
In some cases, we may have a requirement to split a string by other delimiters such as a symbol or specific character. In this example, a string is split using a comma (,) symbol character as a delimiter.

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



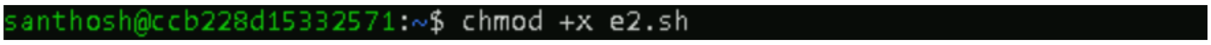
```
santhosh@ccb228d15332571:~$ touch e2.sh
santhosh@ccb228d15332571:~$ nano e2.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



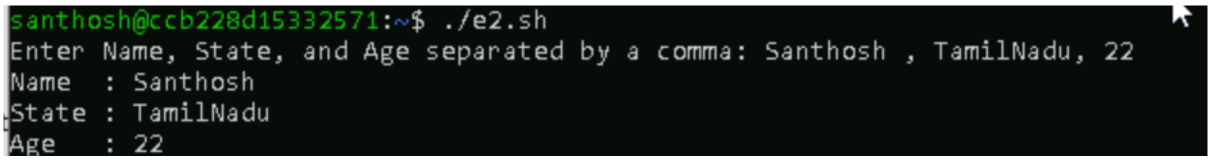
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                                     e2.sh
#!/bin/bash

# Example: Bash split string by symbol (comma)
read -p "Enter Name, State, and Age separated by a comma: " entry

IFS=',' read -r name state age <<< "$entry" # Read input into separate variables

# Trim leading/trailing spaces
name=$(echo "$name" | xargs)
state=$(echo "$state" | xargs)
age=$(echo "$age" | xargs)

# Check if all fields are entered
if [[ -z "$name" || -z "$state" || -z "$age" ]]; then

                                            [ Read 15 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x e2.sh
```

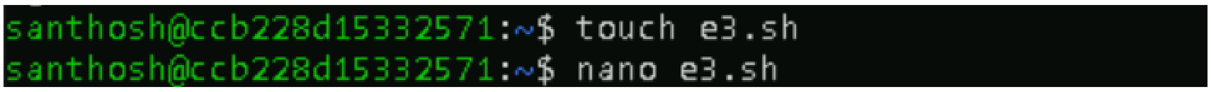
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./e2.sh
Enter Name, State, and Age separated by a comma: Santhosh , TamilNadu, 22
Name  : Santhosh
State : TamilNadu
Age   : 22
```

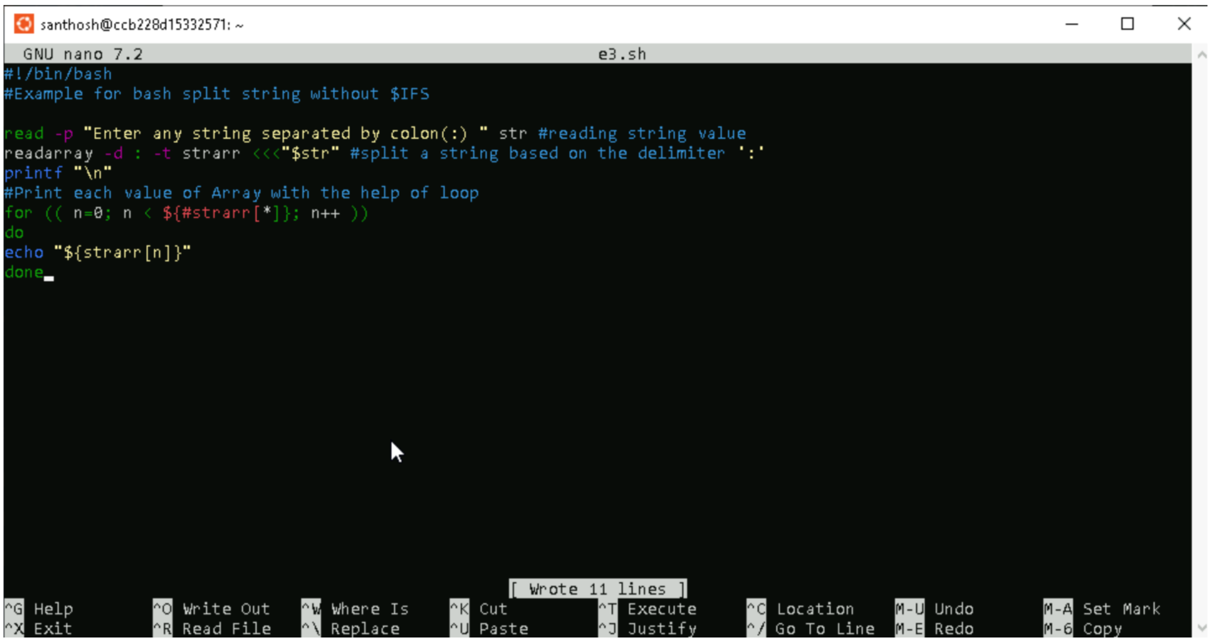
Split without $IFS variable

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



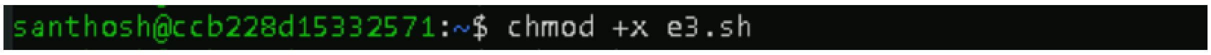
```
santhosh@ccb228d15332571:~$ touch e3.sh
santhosh@ccb228d15332571:~$ nano e3.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



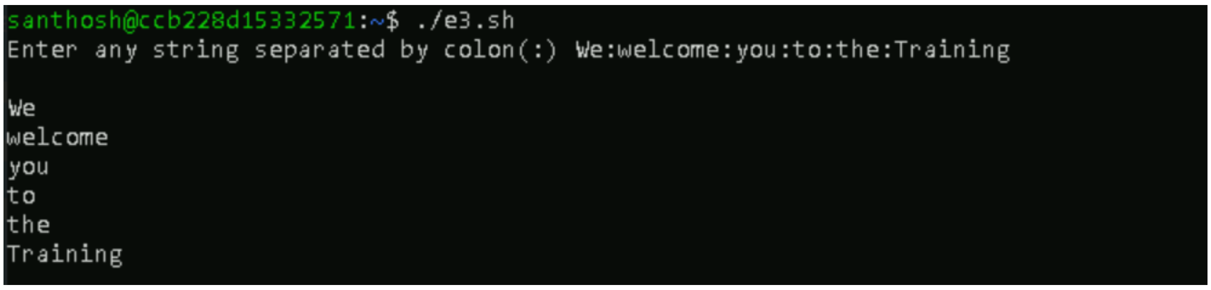
```
#!/bin/bash
#Example for bash split string without $IFS

read -p "Enter any string separated by colon(:) " str #reading string value
readarray -d : -t strarr <<<"$str" #split a string based on the delimiter ':'
printf "\n"
#Print each value of Array with the help of loop
for (( n=0; n < ${#strarr[*]}; n++ ))
do
echo "${strarr[n]}"
done
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x e3.sh
```

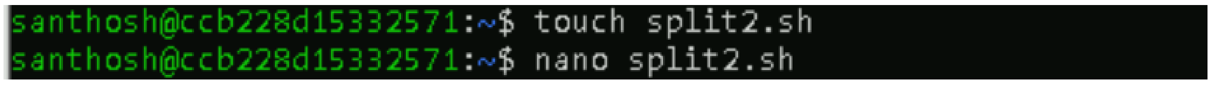
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./e3.sh
Enter any string separated by colon(:) We:welcome:you:to:the:Training

We
welcome
you
to
the
Training
```

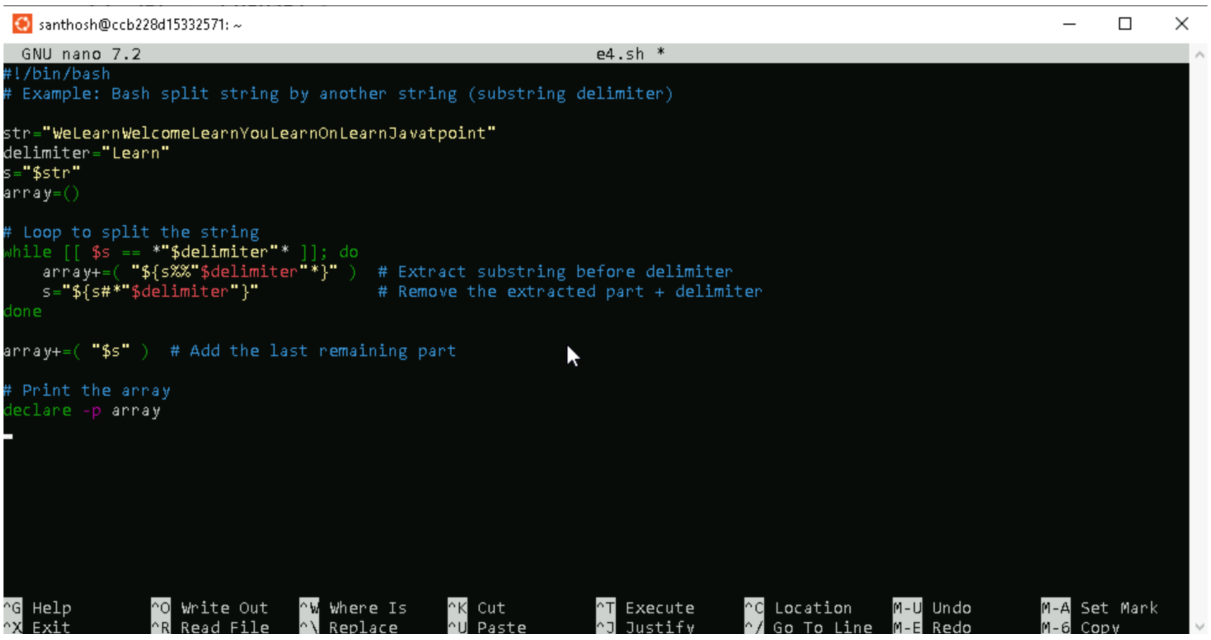
Example 2: Bash Split String by another string.

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



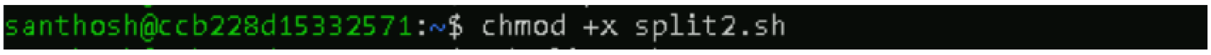
```
santhosh@ccb228d15332571:~$ touch split2.sh
santhosh@ccb228d15332571:~$ nano split2.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



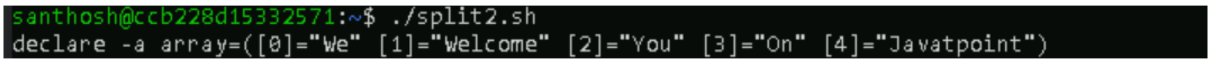
```
santhosh@ccb228d15332571: ~
 GNU nano 7.2                                          e4.sh *
#!/bin/bash
# Example: Bash split string by another string (substring delimiter)

str="WeLearnWelcomeLearnYouLearnOnLearnJavatpoint"
delimiter="Learn"
s="$str"
array=()

# Loop to split the string
while [[ $s == *"$delimiter"* ]]; do
    array+=( "${s%%"$delimiter"*}" )  # Extract substring before delimiter
    s="${s#*"$delimiter"}"            # Remove the extracted part + delimiter
done

array+=( "$s" )  # Add the last remaining part

# Print the array
declare -p array

^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x split2.sh
```

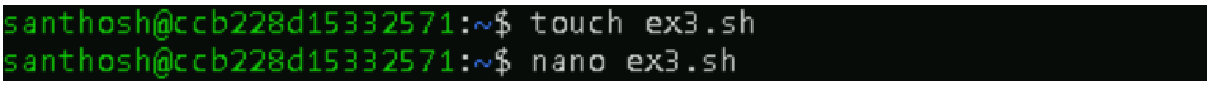
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./split2.sh
declare -a array=([0]="We" [1]="Welcome" [2]="You" [3]="On" [4]="Javatpoint")
```

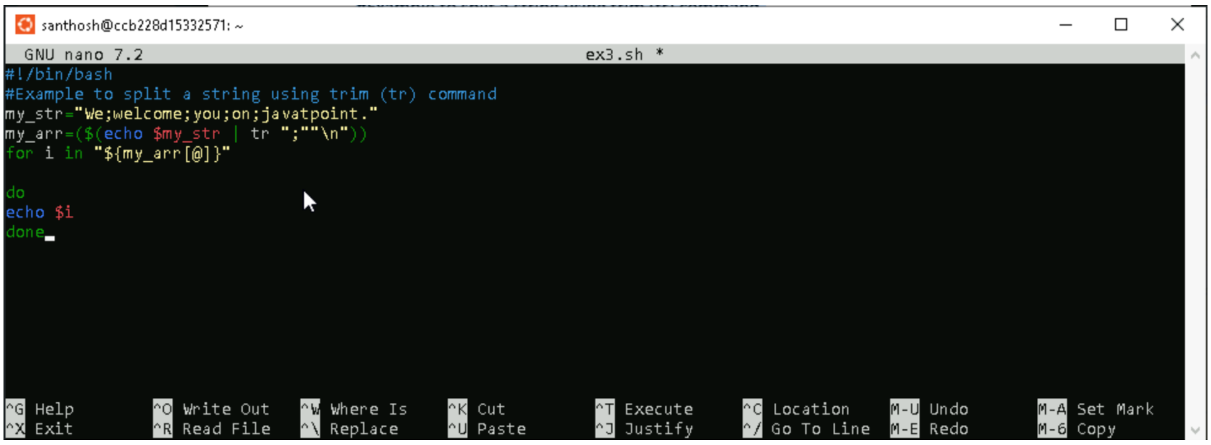
Example 3: Bash Split String using Trim Command

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



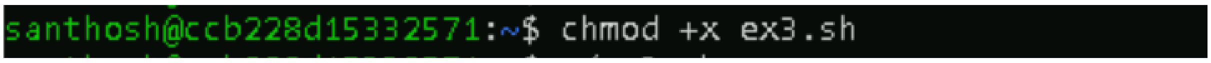
```
santhosh@ccb228d15332571:~$ touch ex3.sh
santhosh@ccb228d15332571:~$ nano ex3.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



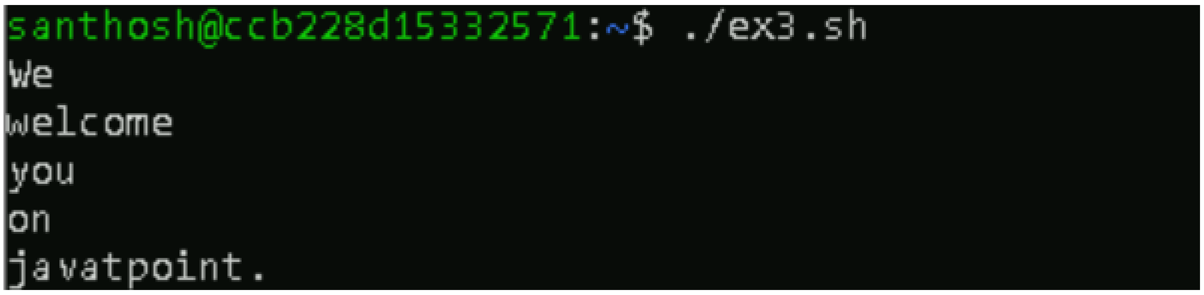
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                                  ex3.sh *
#!/bin/bash
#Example to split a string using trim (tr) command
my_str="We;welcome;you;on;javatpoint."
my_arr=($(echo $my_str | tr ";""\n"))
for i in "${my_arr[@]}"

do
echo $i
done
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x ex3.sh
```

Step 4: Executing the output.



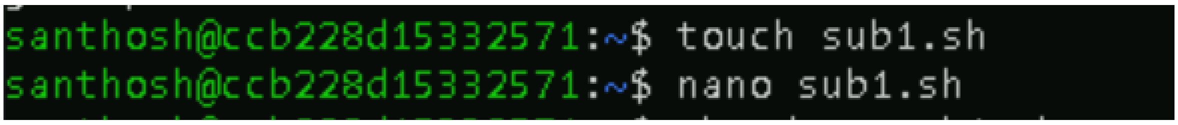
```
santhosh@ccb228d15332571:~$ ./ex3.sh
We
welcome
you
on
javatpoint.
```

Bash Substring

A substring is a sequence of characters within a string. Bash provides an option to extract the information from a string itself. You can extract the digits or a given string using several methods.

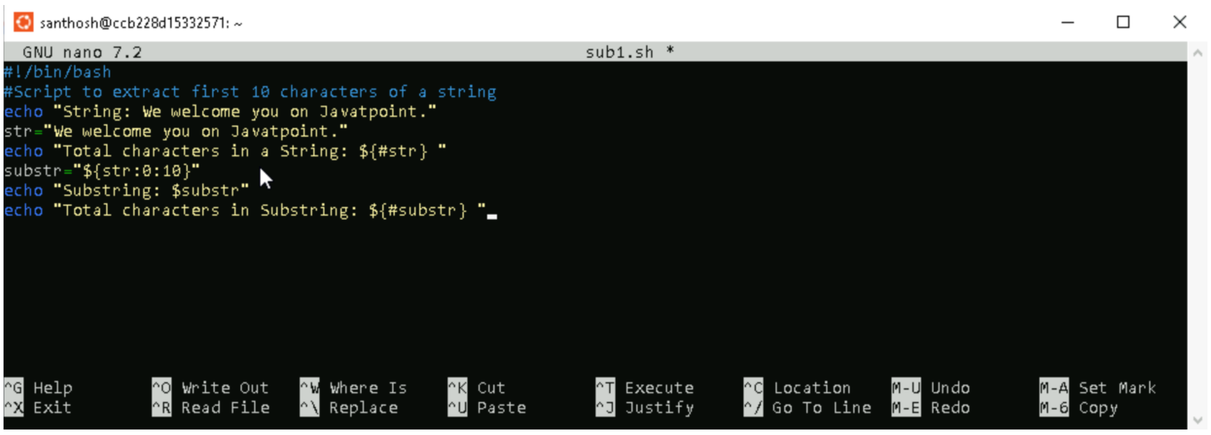
Example 1: To Extract till Specific Characters from Starting

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



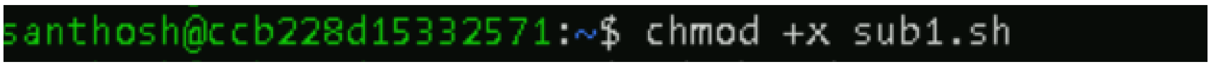
```
santhosh@ccb228d15332571:~$ touch sub1.sh
santhosh@ccb228d15332571:~$ nano sub1.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



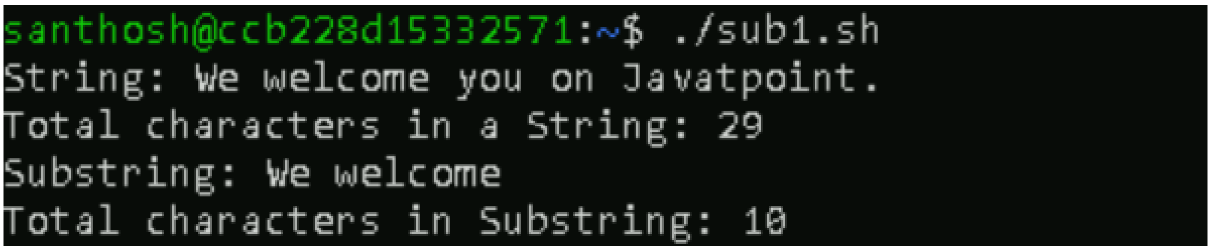
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                          sub1.sh *
#!/bin/bash
#Script to extract first 10 characters of a string
echo "String: We welcome you on Javatpoint."
str="We welcome you on Javatpoint."
echo "Total characters in a String: ${#str} "
substr="${str:0:10}"
echo "Substring: $substr"
echo "Total characters in Substring: ${#substr} "
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x sub1.sh
```

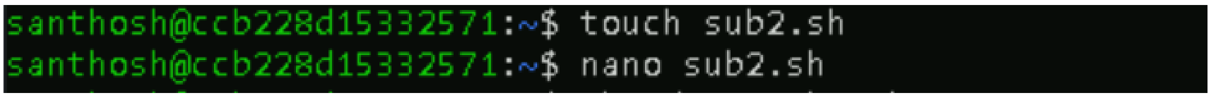
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./sub1.sh
String: We welcome you on Javatpoint.
Total characters in a String: 29
Substring: We welcome
Total characters in Substring: 10
```

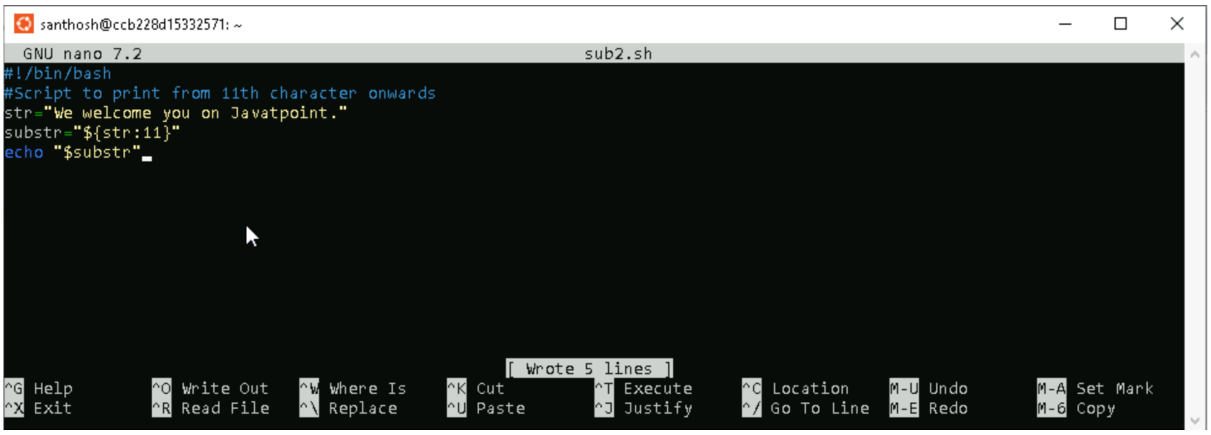
Example 2: To Extract from Specific Character onwards

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



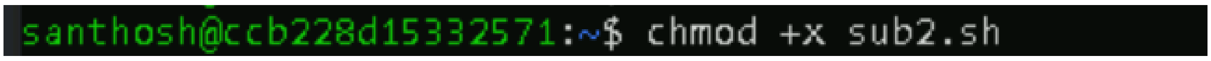
```
santhosh@ccb228d15332571:~$ touch sub2.sh
santhosh@ccb228d15332571:~$ nano sub2.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



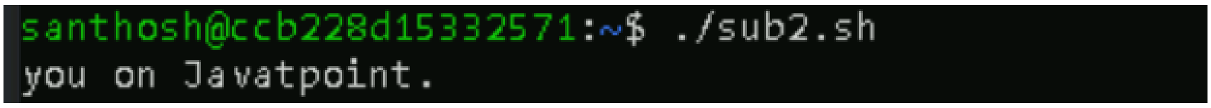
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                                  sub2.sh
#!/bin/bash
#Script to print from 11th character onwards
str="We welcome you on Javatpoint."
substr="${str:11}"
echo "$substr"

                                            [ Wrote 5 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x sub2.sh
```

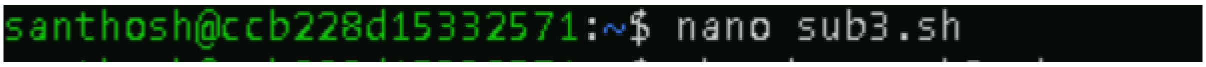
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./sub2.sh
you on Javatpoint.
```

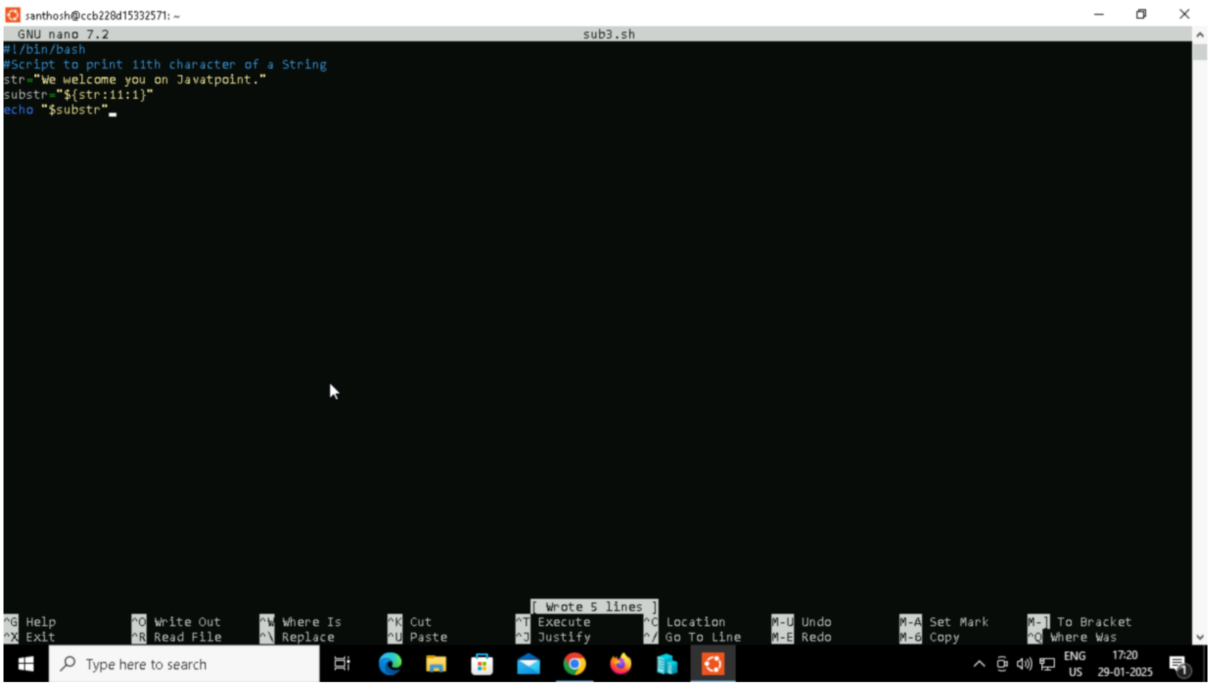
Example 3: To Extract a Single Character

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



```
santhosh@ccb228d15332571:~$ nano sub3.sh
```

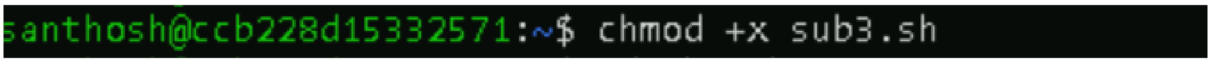
Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



```
GNU nano 7.2                                   sub3.sh
#!/bin/bash
#Script to print 11th character of a String
str="We welcome you on Javatpoint."
substr="${str:11:1}"
echo "$substr"
```

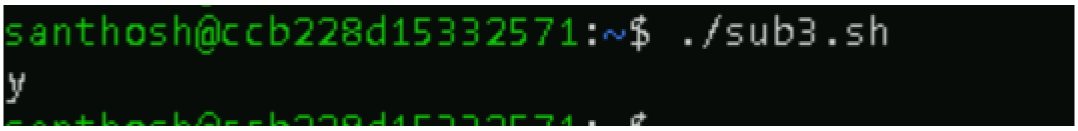
```
                                           [ Wrote 5 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark   M-] To Bracket
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy       ^Q Where Was
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x sub3.sh
```

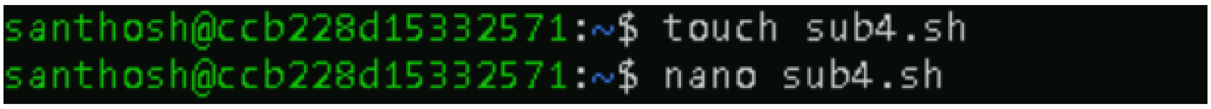
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./sub3.sh
y
```

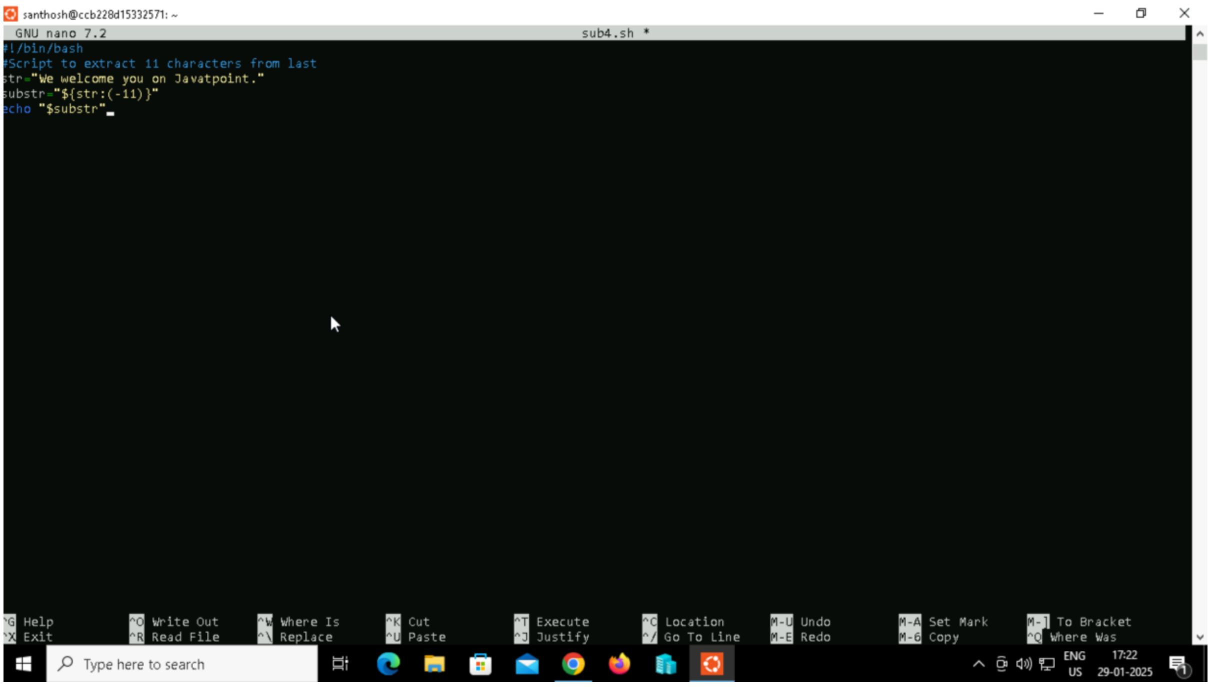
Example 4: To Extract the specific characters from last

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



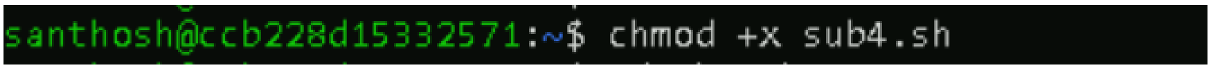
```
santhosh@ccb228d15332571:~$ touch sub4.sh
santhosh@ccb228d15332571:~$ nano sub4.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



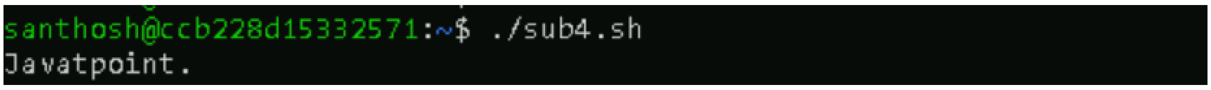
```
#!/bin/bash
#Script to extract 11 characters from last
str="We welcome you on Javatpoint."
substr="${str:(-11)}"
echo "$substr"
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x sub4.sh
```

Step 4: Executing the output.



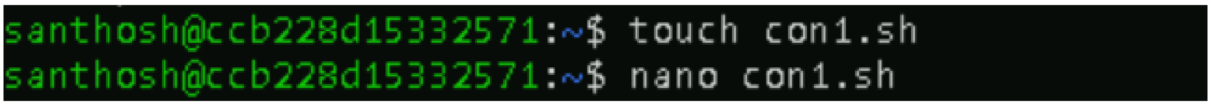
```
santhosh@ccb228d15332571:~$ ./sub4.sh
Javatpoint.
```

Bash Concatenate String

In bash scripting, we can add or join two or more strings together, which is known as string concatenation. It is one of the common requirement for any programming language. A special character or built-in function is applied to perform string concatenation. However, Bash does not contain any built-in function to combine string data or variables. The easiest method to perform string concatenation in bash is to write variables side by side.

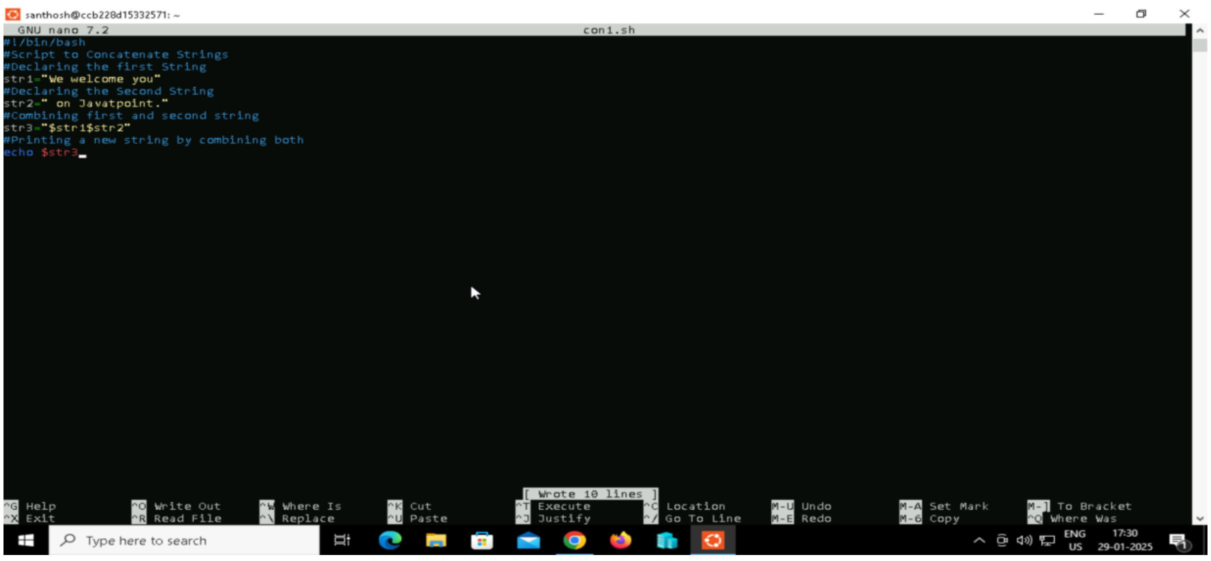
Example 1: Write Variables Side by Side

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



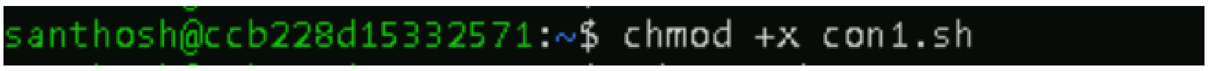
```
santhosh@ccb228d15332571:~$ touch con1.sh
santhosh@ccb228d15332571:~$ nano con1.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



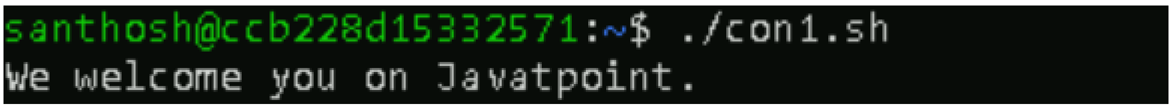
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                                    con1.sh
#!/bin/bash
#Script to Concatenate Strings
#Declaring the first String
str1="We welcome you"
#Declaring the Second String
str2=" on Javatpoint."
#Combining first and second string
str3="$str1$str2"
#Printing a new string by combining both
echo $str3

                                    [ Wrote 10 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark  M-] To Bracket
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy      ^Q Where Was
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x con1.sh
```

Step 4: Executing the output**.**

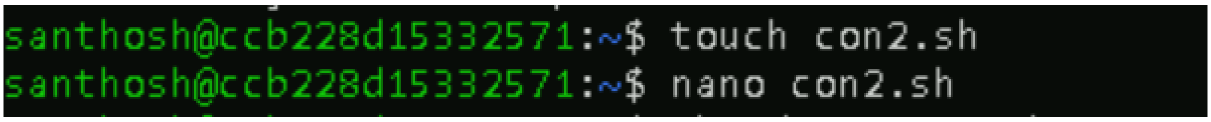


```
santhosh@ccb228d15332571:~$ ./con1.sh
We welcome you on Javatpoint.
```

Example 2: Using Double Quotes

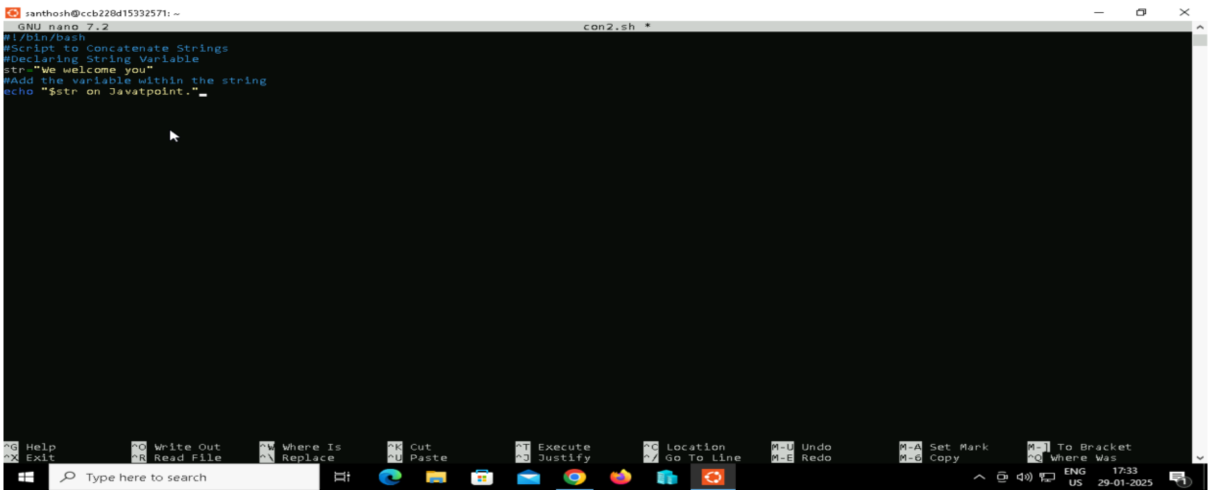
Another easy method is to use variables inside the string, which is defined with double-quotes. The string variable can be applied in any position of the string data**.**

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



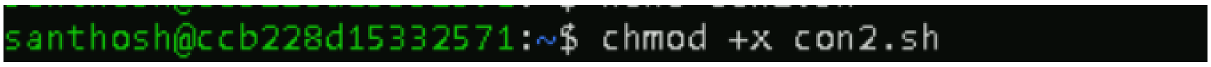
```
santhosh@ccb228d15332571:~$ touch con2.sh
santhosh@ccb228d15332571:~$ nano con2.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



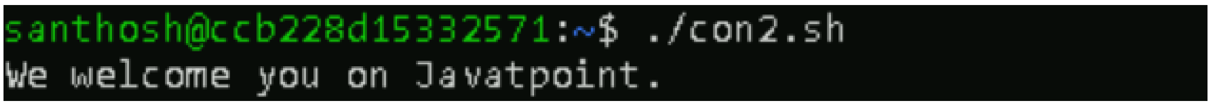
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                                con2.sh *
#!/bin/bash
#Script to Concatenate Strings
#Declaring String Variable
str="We welcome you"
#Add the variable within the string
echo "$str on Javatpoint."
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x con2.sh
```

Step 4: Executing the output**.**

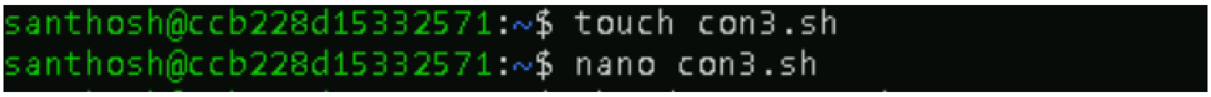


```
santhosh@ccb228d15332571:~$ ./con2.sh
We welcome you on Javatpoint.
```

Example 3: Using Append Operator with Loop

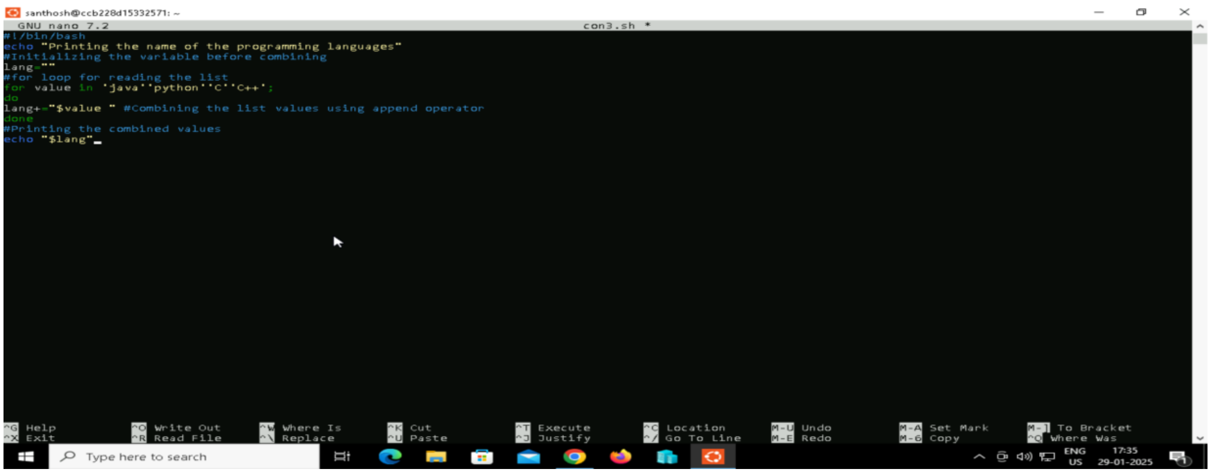
Most of the popular programming languages provide support for append operator (+=) which is the combination of the plus and equal sign. It will add new strings to the end of the string variable.

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



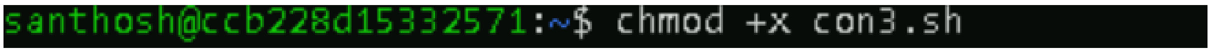
```
santhosh@ccb228d15332571:~$ touch con3.sh
santhosh@ccb228d15332571:~$ nano con3.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



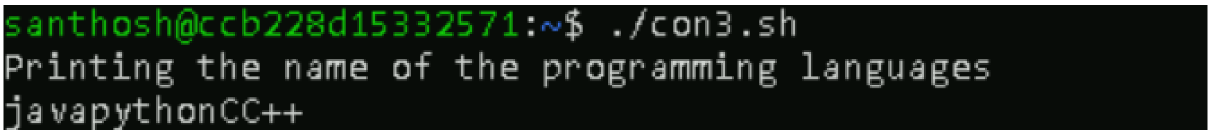
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                                   con3.sh *
#!/bin/bash
echo "Printing the name of the programming languages"
#Initializing the variable before combining
lang=""
#for loop for reading the list
for value in 'java''python''C''C++';
do
lang+="$value " #Combining the list values using append operator
done
#Printing the combined values
echo "$lang"
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x con3.sh
```

Step 4: Executing the output**.**

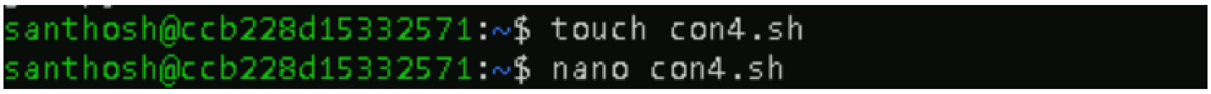


```
santhosh@ccb228d15332571:~$ ./con3.sh
Printing the name of the programming languages
javapythonCC++
```

Example 4: Using the Printf Function

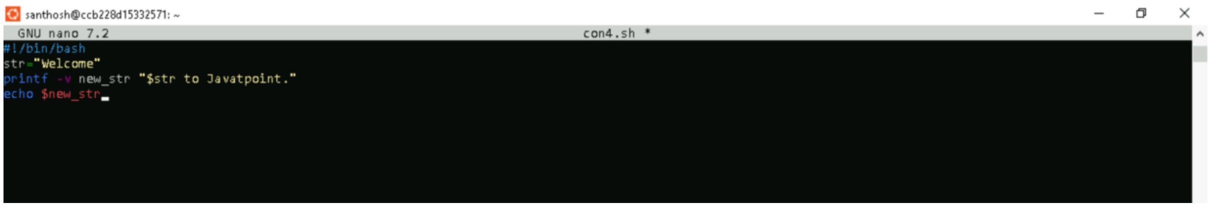
In bash, printf is a function which is used to print and concatenate the strings.

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



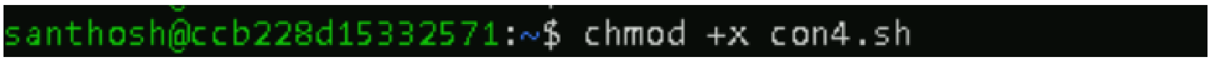
```
santhosh@ccb228d15332571:~$ touch con4.sh
santhosh@ccb228d15332571:~$ nano con4.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



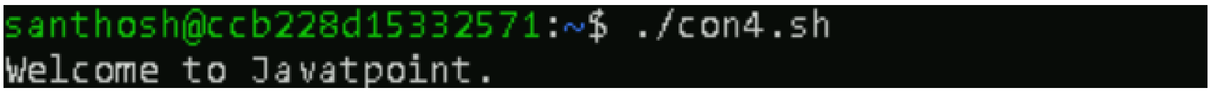
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                         con4.sh *
#!/bin/bash
str="Welcome"
printf -v new_str "$str to Javatpoint."
echo $new_str
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x con4.sh
```

Step 4: Executing the output**.**

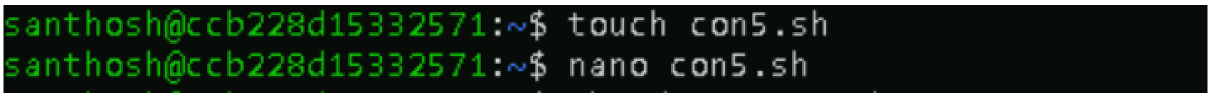


```
santhosh@ccb228d15332571:~$ ./con4.sh
Welcome to Javatpoint.
```

Example 5: Using Literal Strings

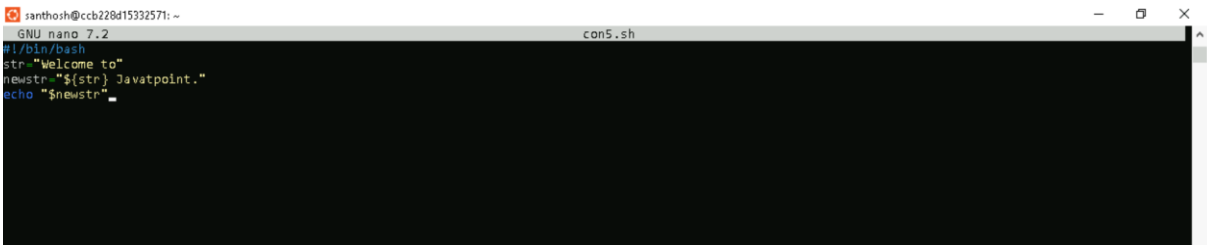
String concatenation can also be performed with a literal string by using curly braces{}. They should be used in such a way that the variable does not mix up with the literal string.

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



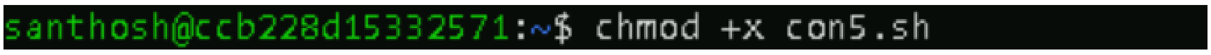
```
santhosh@ccb228d15332571:~$ touch con5.sh
santhosh@ccb228d15332571:~$ nano con5.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



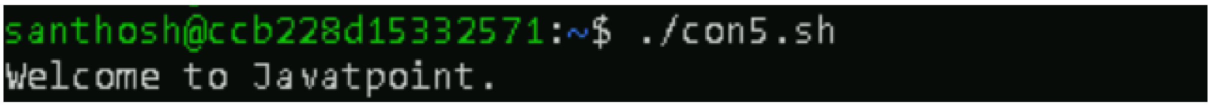
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                                con5.sh
#!/bin/bash
str="Welcome to"
newstr="${str} Javatpoint."
echo "$newstr"
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x con5.sh
```

Step 4: Executing the output**.**

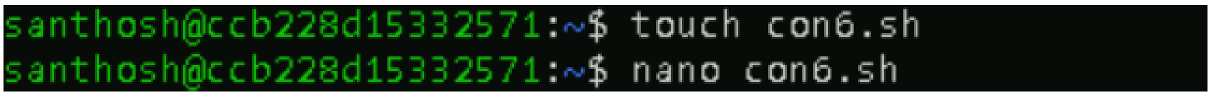


```
santhosh@ccb228d15332571:~$ ./con5.sh
Welcome to Javatpoint.
```

Example 6: Using Underscore Using underscore

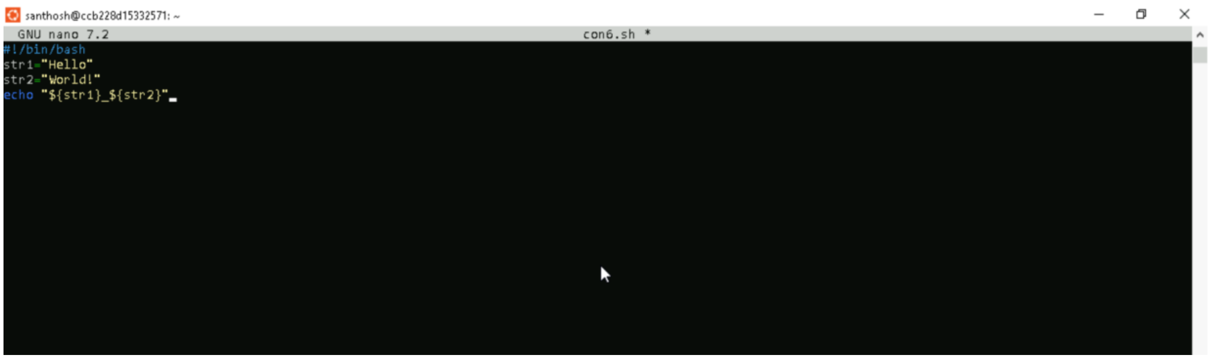
For concatenating the string in bash shell is one of the common tasks. It is mostly used for assigning a name to the files.

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



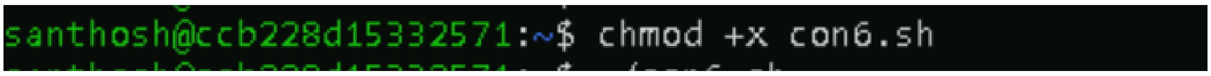
```
santhosh@ccb228d15332571:~$ touch con6.sh
santhosh@ccb228d15332571:~$ nano con6.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



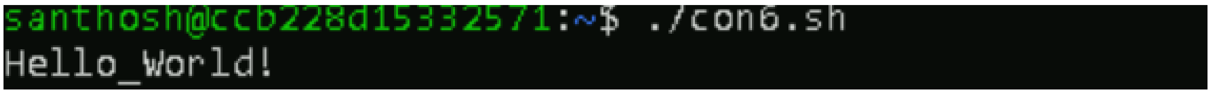
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                                con6.sh *
#!/bin/bash
str1="Hello"
str2="World!"
echo "${str1}_${str2}"
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x con6.sh
```

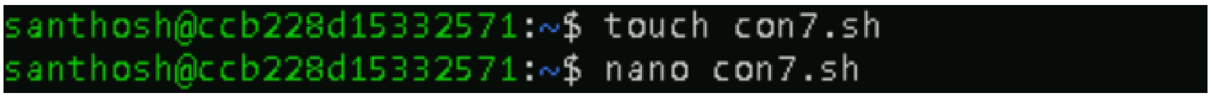
Step 4: Executing the output**.**



```
santhosh@ccb228d15332571:~$ ./con6.sh
Hello_World!
```

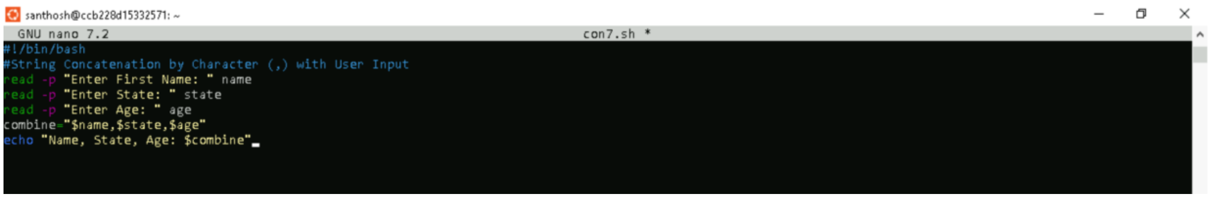
Example 7: Using any Character

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



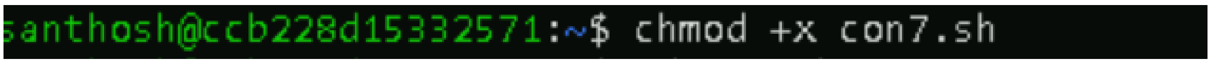
```
santhosh@ccb228d15332571:~$ touch con7.sh
santhosh@ccb228d15332571:~$ nano con7.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



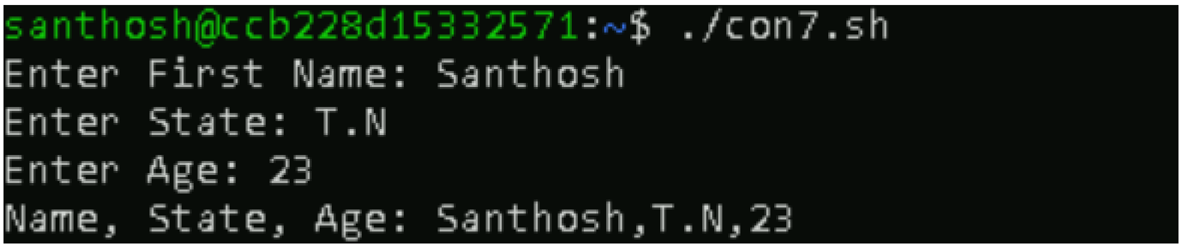
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                          con7.sh *
#!/bin/bash
#String Concatenation by Character (,) with User Input
read -p "Enter First Name: " name
read -p "Enter State: " state
read -p "Enter Age: " age
combine="$name,$state,$age"
echo "Name, State, Age: $combine"
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x con7.sh
```

Step 4: Executing the output**.**

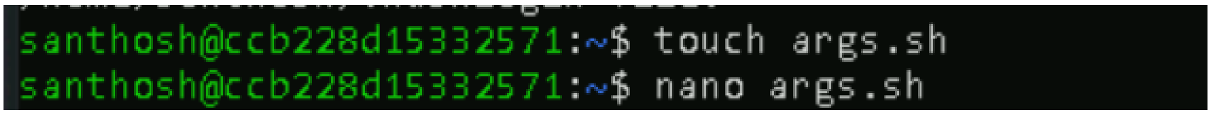


```
santhosh@ccb228d15332571:~$ ./con7.sh
Enter First Name: Santhosh
Enter State: T.N
Enter Age: 23
Name, State, Age: Santhosh,T.N,23
```

Bash Functions

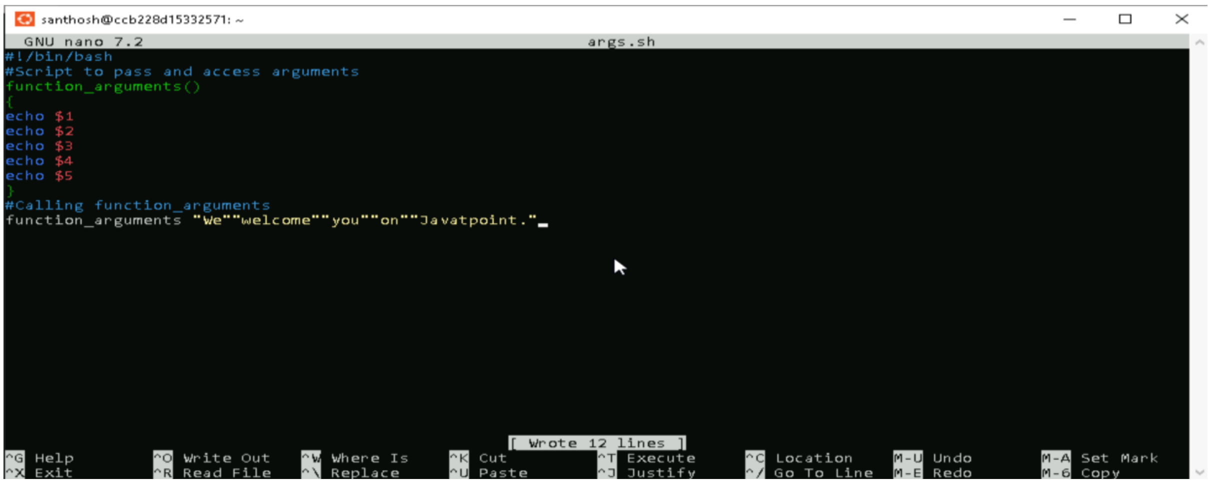
Functions in bash scripting are a great option to reuse code. A Bash function can be defined as a set of commands which can be called several times within bash script. The purpose of function in bash is to help you make your scripts more readable and avoid writing the same code again and again.

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



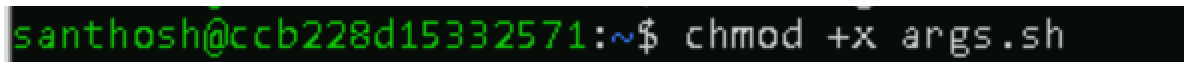
```
santhosh@ccb228d15332571:~$ touch args.sh
santhosh@ccb228d15332571:~$ nano args.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



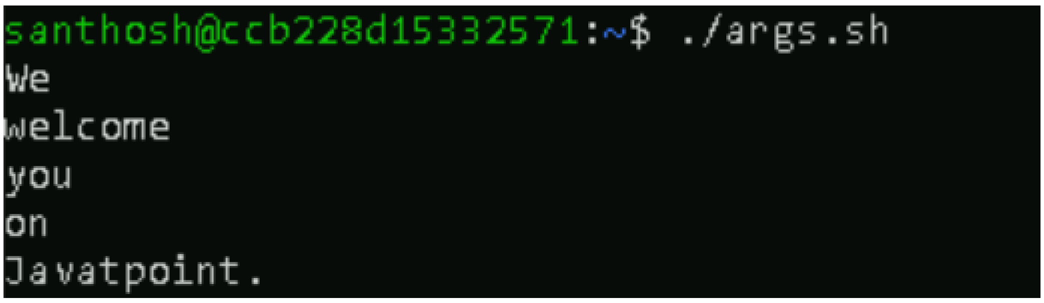
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                                   args.sh
#!/bin/bash
#Script to pass and access arguments
function_arguments()
{
echo $1
echo $2
echo $3
echo $4
echo $5
}
#Calling function_arguments
function_arguments "We""welcome""you""on""Javatpoint."

                                             [ Wrote 12 lines ]
^G Help       ^O Write Out   ^W Where Is   ^K Cut     ^T Execute   ^C Location    M-U Undo   M-A Set Mark
^X Exit       ^R Read File   ^\ Replace    ^U Paste   ^J Justify   ^/ Go To Line  M-E Redo   M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x args.sh
```

Step 4: Executing the output.

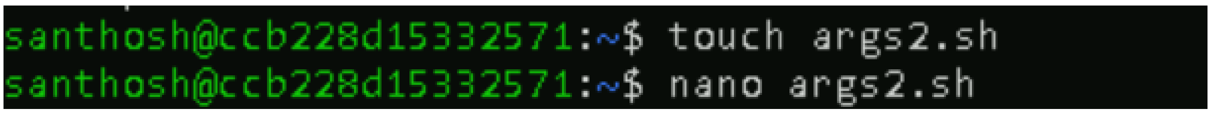


```
santhosh@ccb228d15332571:~$ ./args.sh
We
welcome
you
on
Javatpoint.
```

Variable Scope

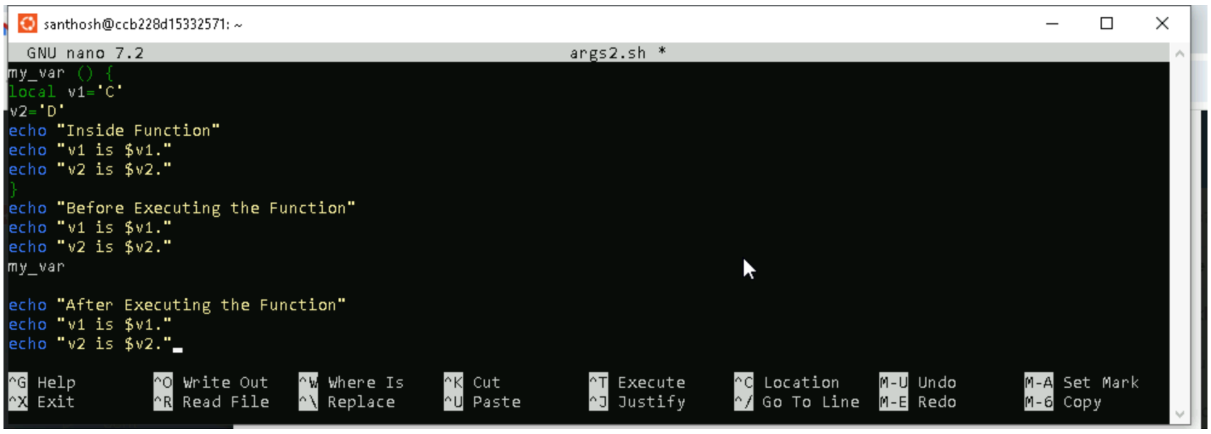
Global variables are defined as the variables which can be accessed anywhere within the script regardless of the scope. By default, all the variables are defined as global variables, even if they are declared inside the function. We can also create variables as a local variable. Local variables can be declared within the function body with the ?local? keyword when they are assigned for first time. They are only accessible inside that function.

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



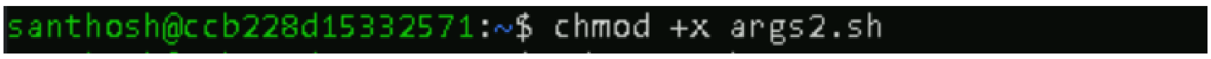
```
santhosh@ccb228d15332571:~$ touch args2.sh
santhosh@ccb228d15332571:~$ nano args2.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



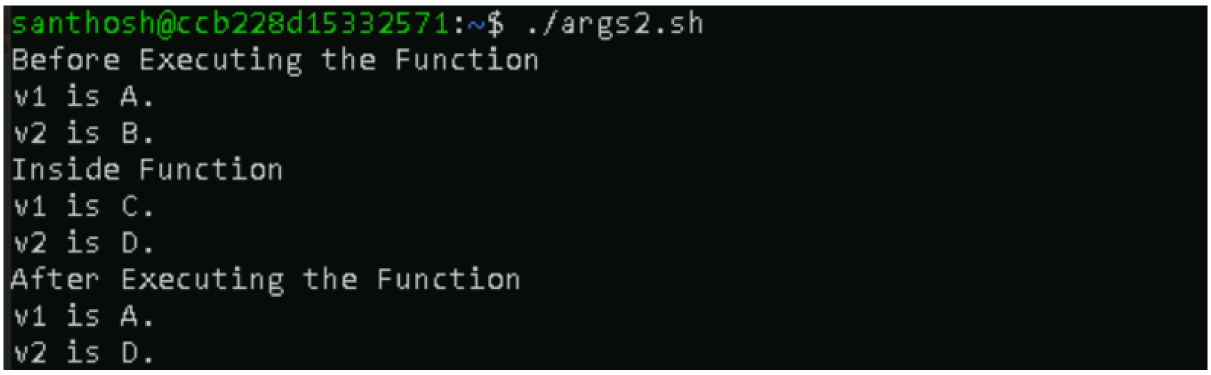
```
  GNU nano 7.2                         args2.sh *
my_var () {
local v1='C'
v2='D'
echo "Inside Function"
echo "v1 is $v1."
echo "v2 is $v2."
}
echo "Before Executing the Function"
echo "v1 is $v1."
echo "v2 is $v2."
my_var

echo "After Executing the Function"
echo "v1 is $v1."
echo "v2 is $v2."
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x args2.sh
```

Step 4: Executing the output.

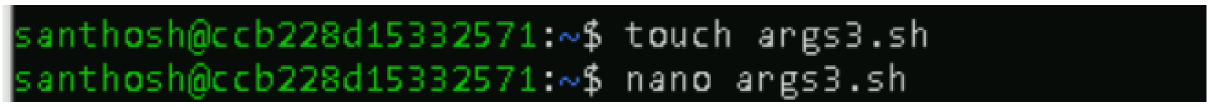


```
santhosh@ccb228d15332571:~$ ./args2.sh
Before Executing the Function
v1 is A.
v2 is B.
Inside Function
v1 is C.
v2 is D.
After Executing the Function
v1 is A.
v2 is D.
```

Return Values

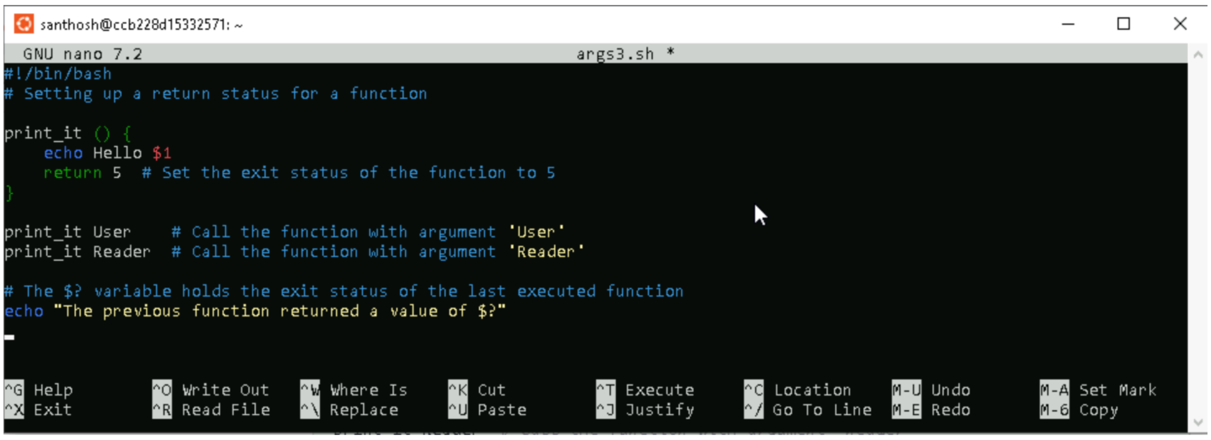
The return status can be indicated by using the 'return' keyword, and it is assigned to the variable $?. The return statement terminates the function and works as the function's exit status.

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



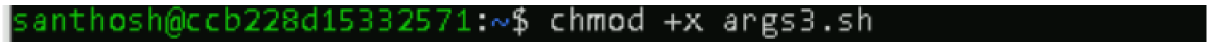
```
santhosh@ccb228d15332571:~$ touch args3.sh
santhosh@ccb228d15332571:~$ nano args3.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



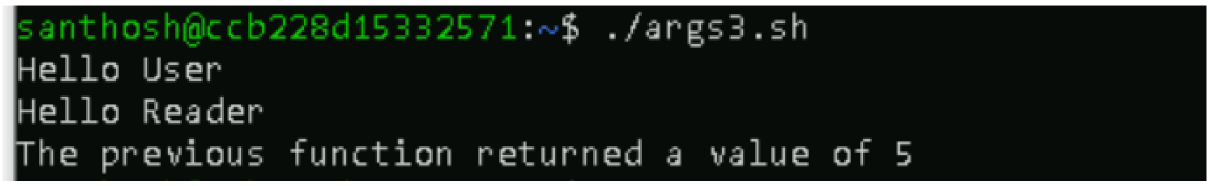
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                          args3.sh *
#!/bin/bash
# Setting up a return status for a function

print_it () {
    echo Hello $1
    return 5  # Set the exit status of the function to 5
}

print_it User    # Call the function with argument 'User'
print_it Reader  # Call the function with argument 'Reader'

# The $? variable holds the exit status of the last executed function
echo "The previous function returned a value of $?"

^G Help      ^O Write Out  ^W Where Is  ^K Cut    ^T Execute  ^C Location   M-U Undo  M-A Set Mark
^X Exit      ^R Read File  ^\ Replace   ^U Paste  ^J Justify  ^/ Go To Line M-E Redo  M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x args3.sh
```

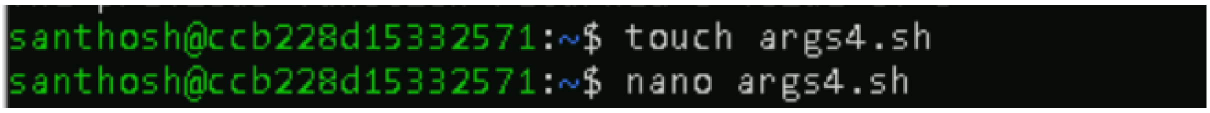
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./args3.sh
Hello User
Hello Reader
The previous function returned a value of 5
```

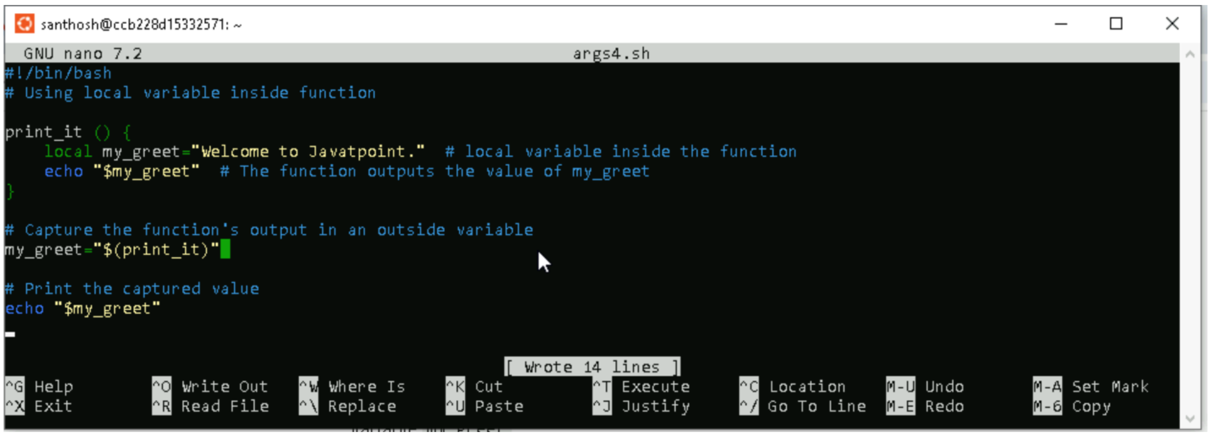
Another better option to return a value from a function is to send the value to stdout using echo or printf commands, as shown below:

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



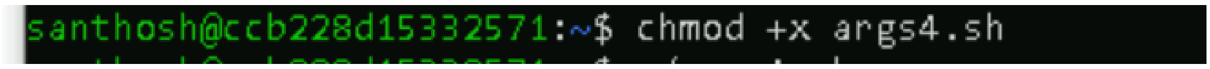
```
santhosh@ccb228d15332571:~$ touch args4.sh
santhosh@ccb228d15332571:~$ nano args4.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



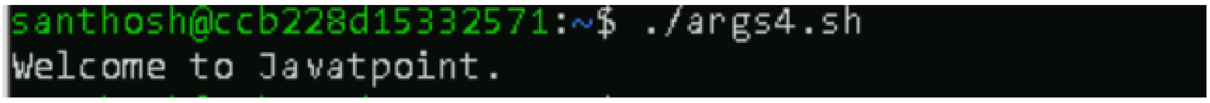
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                                args4.sh
#!/bin/bash
# Using local variable inside function

print_it () {
    local my_greet="Welcome to Javatpoint."  # local variable inside the function
    echo "$my_greet"  # The function outputs the value of my_greet
}

# Capture the function's output in an outside variable
my_greet="$(print_it)"

# Print the captured value
echo "$my_greet"

                                        [ Wrote 14 lines ]
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x args4.sh
```

Step 4: Executing the output.

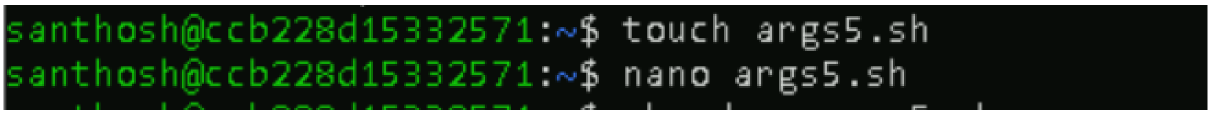


```
santhosh@ccb228d15332571:~$ ./args4.sh
Welcome to Javatpoint.
```

Overriding Commands

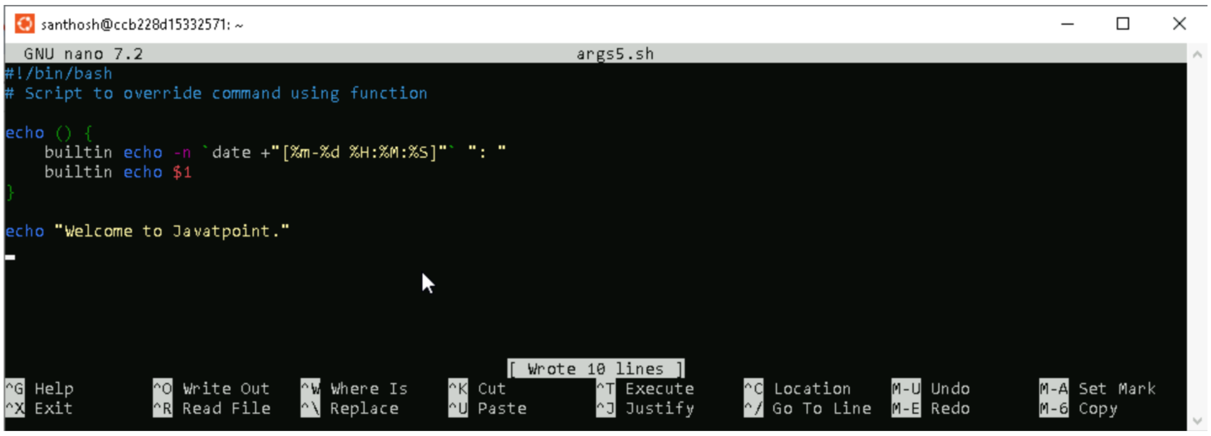
In this example, we have overridden the 'echo' command and added the time stamp in the form of the argument to the 'echo' command.

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



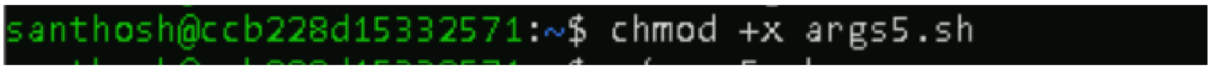
```
santhosh@ccb228d15332571:~$ touch args5.sh
santhosh@ccb228d15332571:~$ nano args5.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



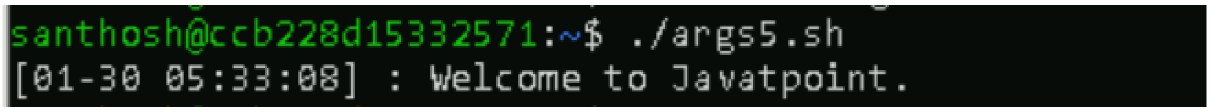
```
#!/bin/bash
# Script to override command using function

echo () {
    builtin echo -n `date +"[%m-%d %H:%M:%S]"` ": "
    builtin echo $1
}

echo "Welcome to Javatpoint."
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x args5.sh
```

Step 4: Executing the output.



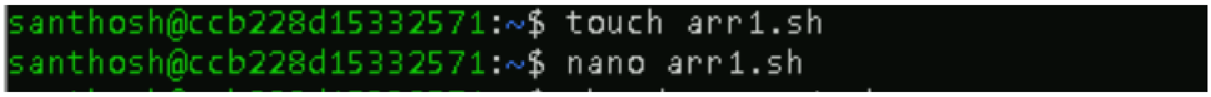
```
santhosh@ccb228d15332571:~$ ./args5.sh
[01-30 05:33:08] : Welcome to Javatpoint.
```

Bash Array

An array can be defined as a collection of similar type of elements. Unlike most of the programming languages, arrays in bash scripting need not be the collection of similar elements. Since Bash does not discriminate the string from a number, an array may contain both strings and numbers.

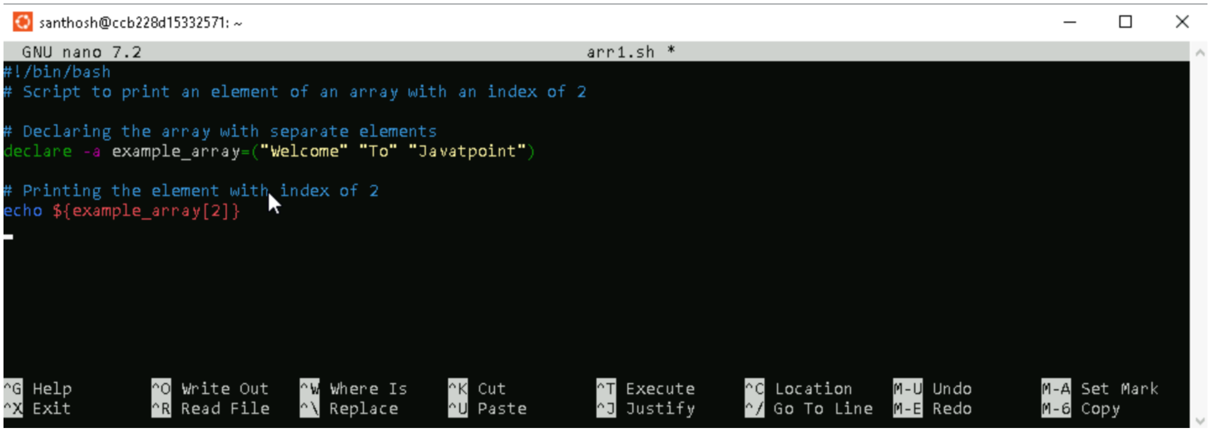
Example 1 : let's print an element of an array with an index of 2:

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



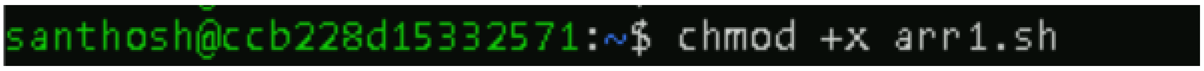
```
santhosh@ccb228d15332571:~$ touch arr1.sh
santhosh@ccb228d15332571:~$ nano arr1.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



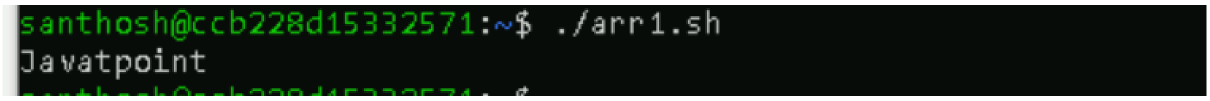
```
  GNU nano 7.2                                    arr1.sh *
#!/bin/bash
# Script to print an element of an array with an index of 2

# Declaring the array with separate elements
declare -a example_array=("Welcome" "To" "Javatpoint")

# Printing the element with index of 2
echo ${example_array[2]}

^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x arr1.sh
```

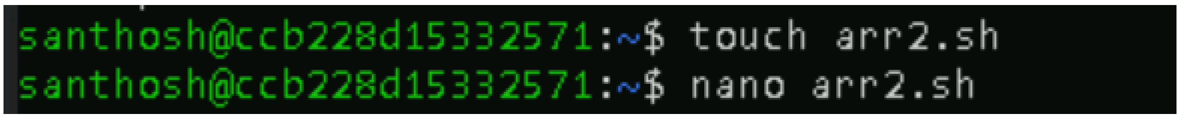
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./arr1.sh
Javatpoint
```

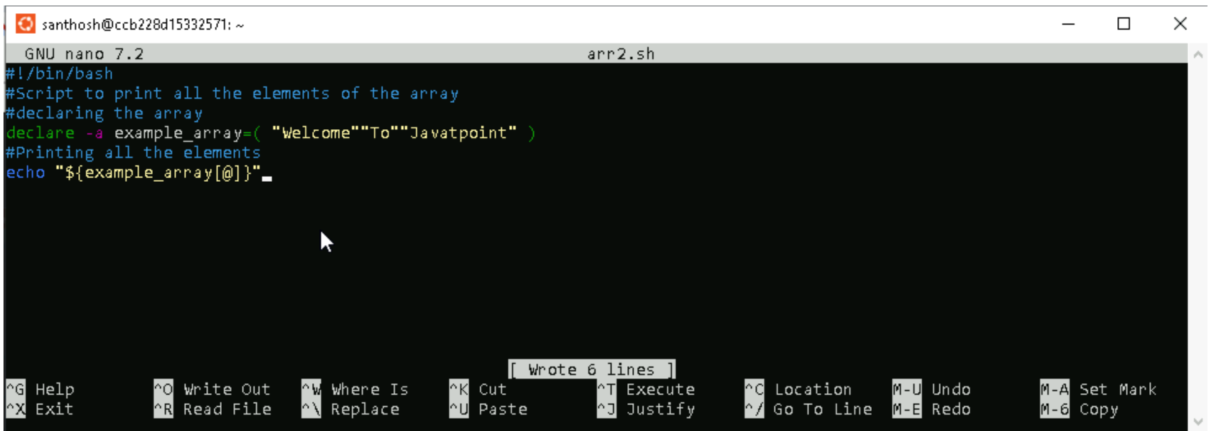
If we use @ or * in the place of a specified index, it will expand to all members of the array. To print all the elements, we can use the following form:

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



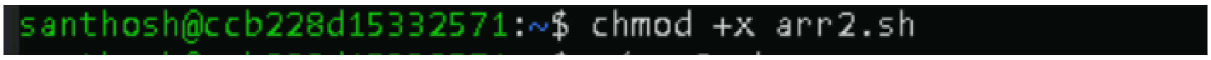
```
santhosh@ccb228d15332571:~$ touch arr2.sh
santhosh@ccb228d15332571:~$ nano arr2.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



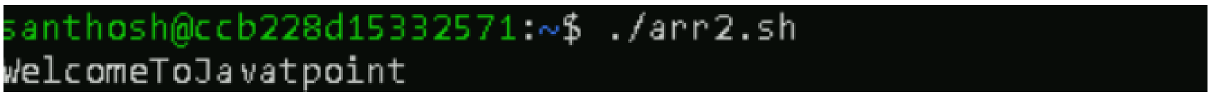
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                              arr2.sh
#!/bin/bash
#Script to print all the elements of the array
#declaring the array
declare -a example_array=( "Welcome""To""Javatpoint" )
#Printing all the elements
echo "${example_array[@]}"

                                 [ Wrote 6 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x arr2.sh
```

Step 4: Executing the output.



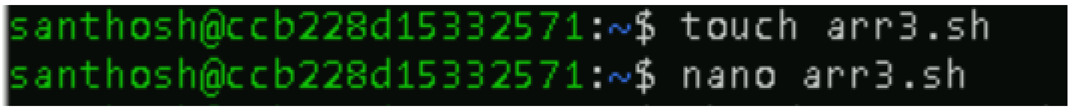
```
santhosh@ccb228d15332571:~$ ./arr2.sh
WelcomeToJavatpoint
```

Printing the Keys of an Array

We can also retrieve and print the keys used in indexed or associative arrays, instead of their respective values. It can be performed by adding the ! operator before the array name as below:

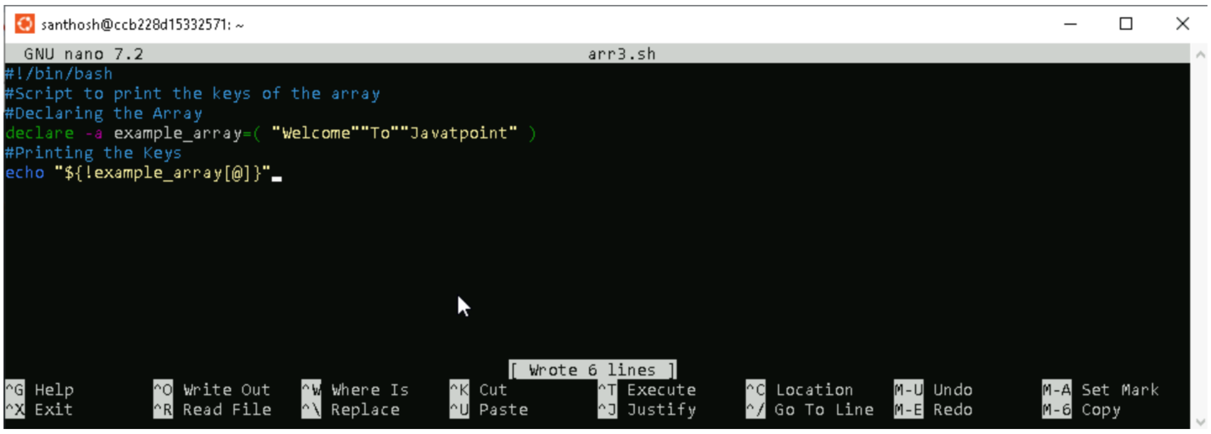
1. ${!ARRAY_NAME[index]}

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



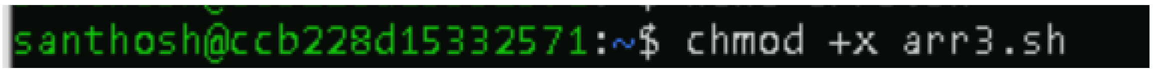
```
santhosh@ccb228d15332571:~$ touch arr3.sh
santhosh@ccb228d15332571:~$ nano arr3.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



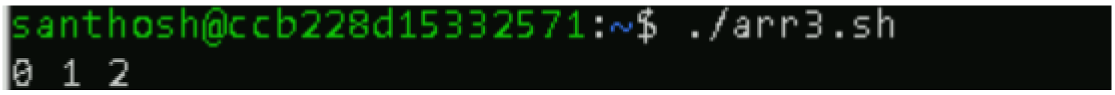
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                                arr3.sh
#!/bin/bash
#Script to print the keys of the array
#Declaring the Array
declare -a example_array=( "Welcome""To""Javatpoint" )
#Printing the Keys
echo "${!example_array[@]}"

                                   [ Wrote 6 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x arr3.sh
```

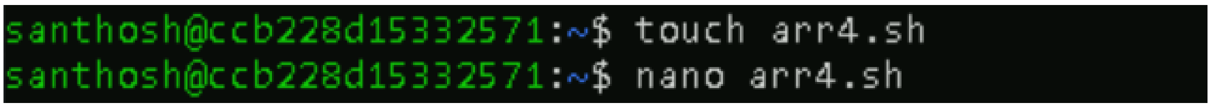
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./arr3.sh
0 1 2
```

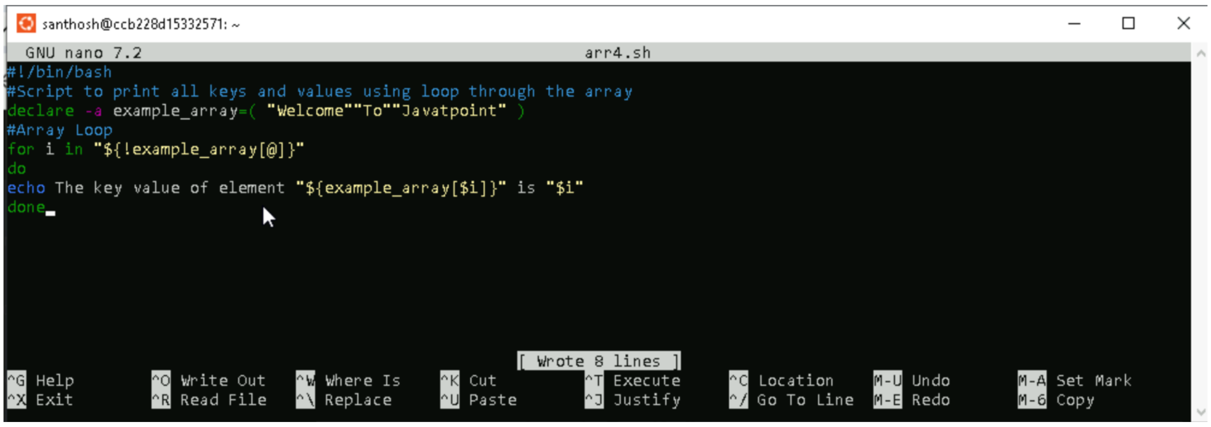
Loop through the Array

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



```
santhosh@ccb228d15332571:~$ touch arr4.sh
santhosh@ccb228d15332571:~$ nano arr4.sh
```

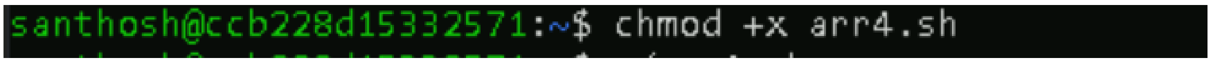
Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                          arr4.sh
#!/bin/bash
#Script to print all keys and values using loop through the array
declare -a example_array=( "Welcome""To""Javatpoint" )
#Array Loop
for i in "${!example_array[@]}"
do
echo The key value of element "${example_array[$i]}" is "$i"
done
```

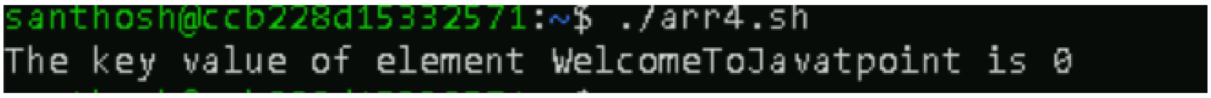
```
                                   [ Wrote 8 lines ]
^G Help       ^O Write Out   ^W Where Is    ^K Cut        ^T Execute    ^C Location    M-U Undo      M-A Set Mark
^X Exit       ^R Read File   ^\ Replace     ^U Paste      ^J Justify    ^/ Go To Line  M-E Redo      M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x arr4.sh
```

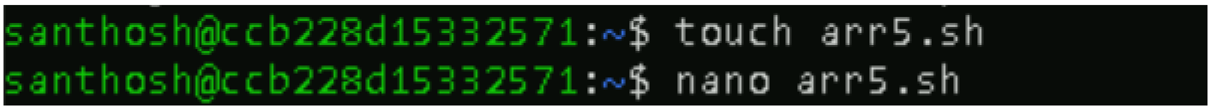
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./arr4.sh
The key value of element WelcomeToJavatpoint is 0
```

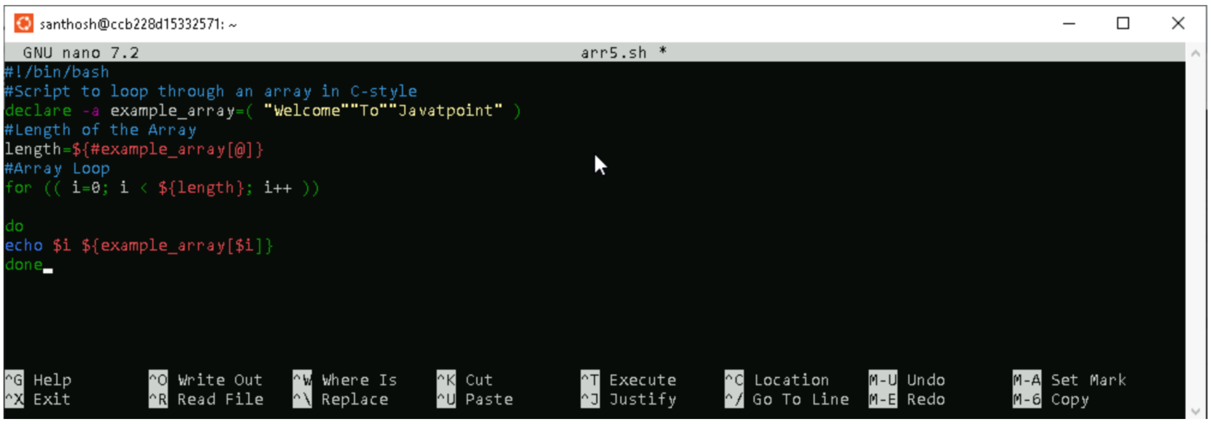
Another common method to loop through an array is to retrieve the length of the array and use the C-style loop:

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



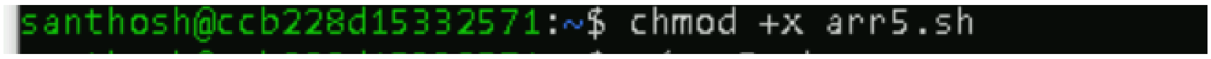
```
santhosh@ccb228d15332571:~$ touch arr5.sh
santhosh@ccb228d15332571:~$ nano arr5.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



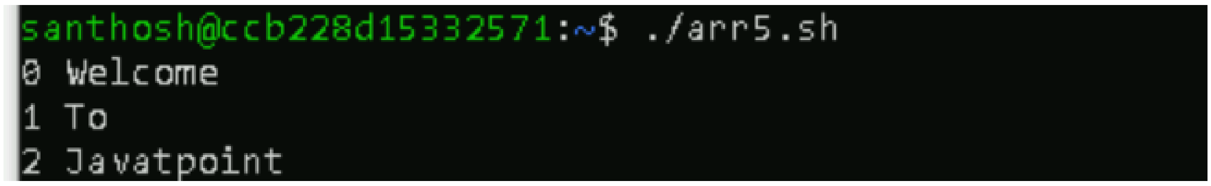
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                                   arr5.sh *
#!/bin/bash
#Script to loop through an array in C-style
declare -a example_array=( "Welcome""To""Javatpoint" )
#Length of the Array
length=${#example_array[@]}
#Array Loop
for (( i=0; i < ${length}; i++ ))

do
echo $i ${example_array[$i]}
done
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x arr5.sh
```

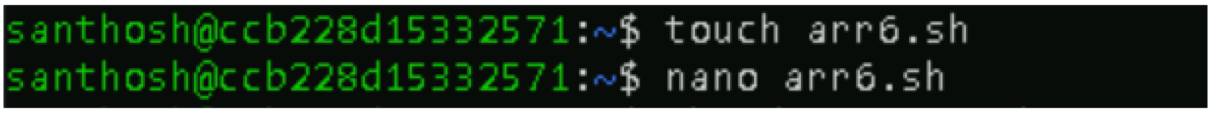
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./arr5.sh
0 Welcome
1 To
2 Javatpoint
```

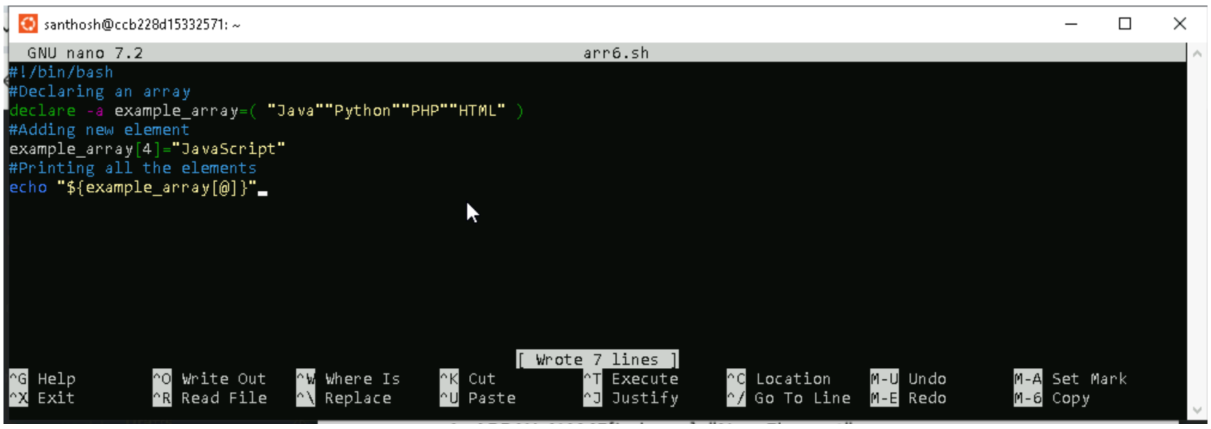
Adding Elements to an Array

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



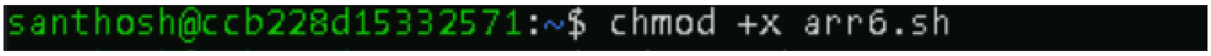
```
santhosh@ccb228d15332571:~$ touch arr6.sh
santhosh@ccb228d15332571:~$ nano arr6.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



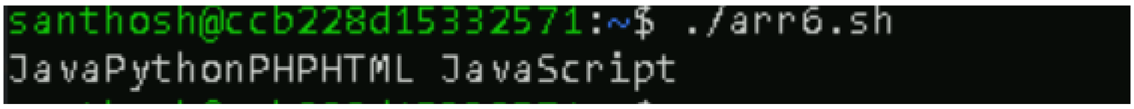
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                              arr6.sh
#!/bin/bash
#Declaring an array
declare -a example_array=( "Java""Python""PHP""HTML" )
#Adding new element
example_array[4]="JavaScript"
#Printing all the elements
echo "${example_array[@]}"

                                  [ Wrote 7 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x arr6.sh
```

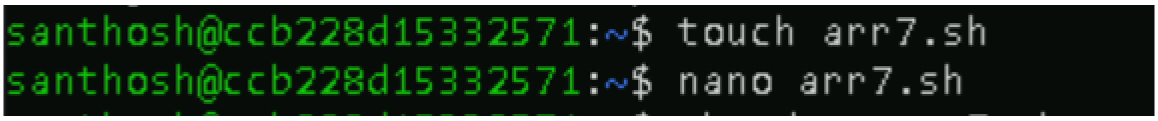
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./arr6.sh
JavaPythonPHPHTML JavaScript
```

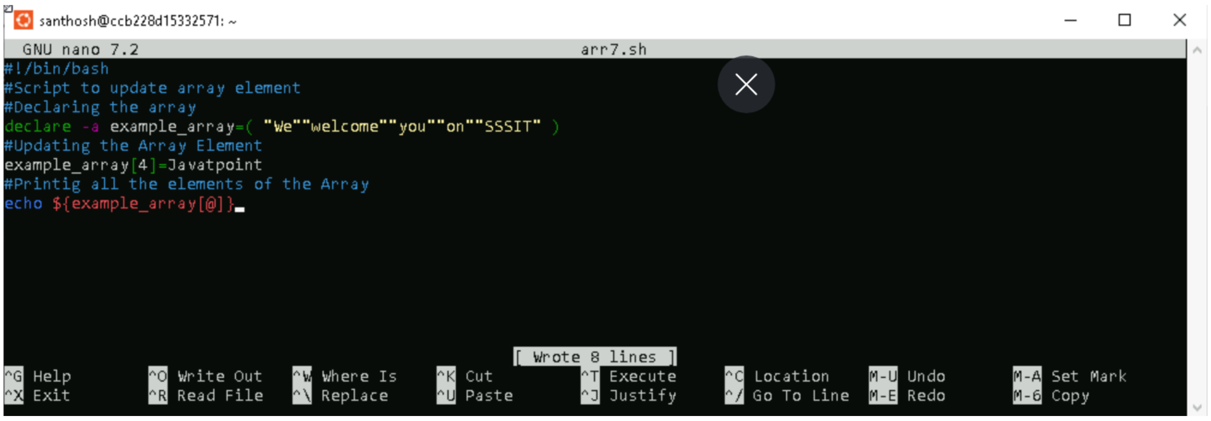
Updating Array Element

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



```
santhosh@ccb228d15332571:~$ touch arr7.sh
santhosh@ccb228d15332571:~$ nano arr7.sh
```

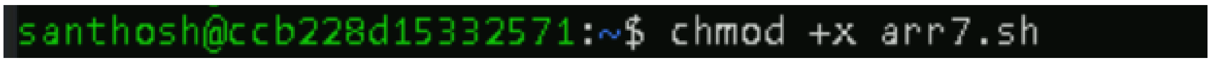
Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



```
#!/bin/bash
#Script to update array element
#Declaring the array
declare -a example_array=( "We""welcome""you""on""SSSIT" )
#Updating the Array Element
example_array[4]=Javatpoint
#Printig all the elements of the Array
echo ${example_array[@]}
```

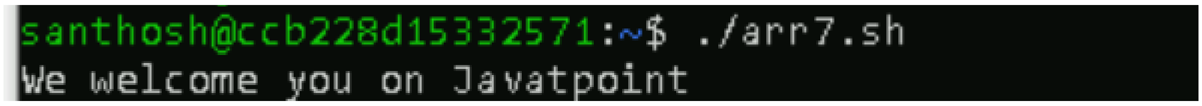
```
[ Wrote 8 lines ]
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x arr7.sh
```

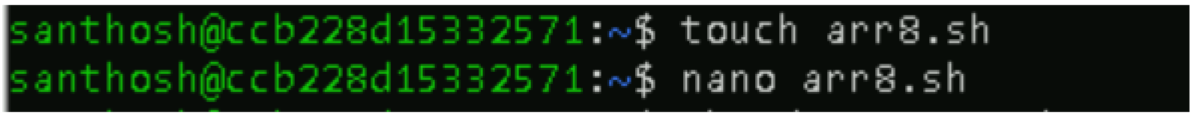
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./arr7.sh
We welcome you on Javatpoint
```

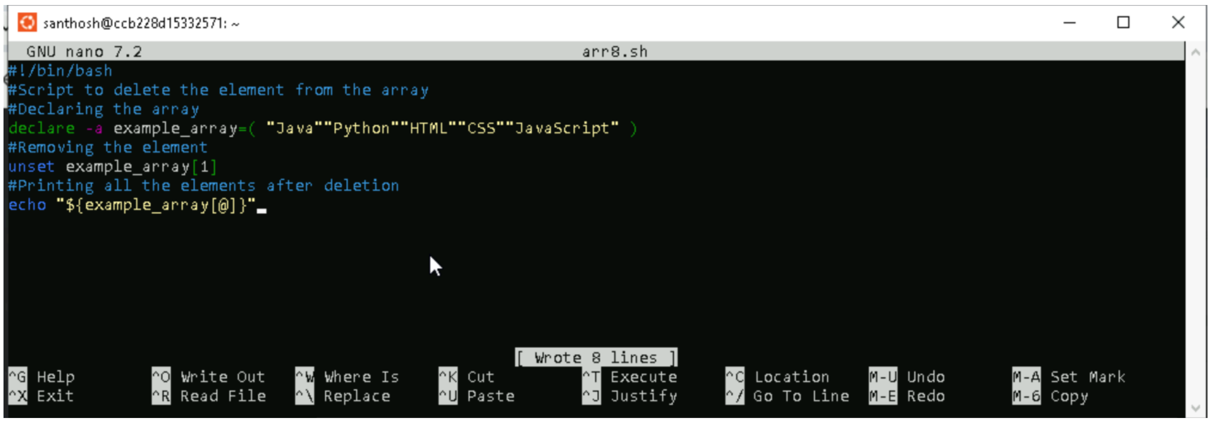
Deleting an Element from an Array

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



```
santhosh@ccb228d15332571:~$ touch arr8.sh
santhosh@ccb228d15332571:~$ nano arr8.sh
```

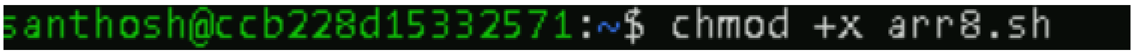
Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



```
#!/bin/bash
#Script to delete the element from the array
#Declaring the array
declare -a example_array=( "Java""Python""HTML""CSS""JavaScript" )
#Removing the element
unset example_array[1]
#Printing all the elements after deletion
echo "${example_array[@]}"
```

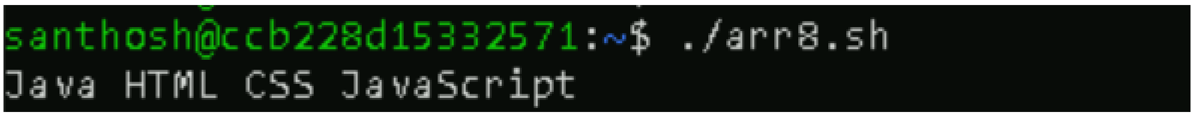
```
                                        [ Wrote 8 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x arr8.sh
```

Step 4: Executing the output.

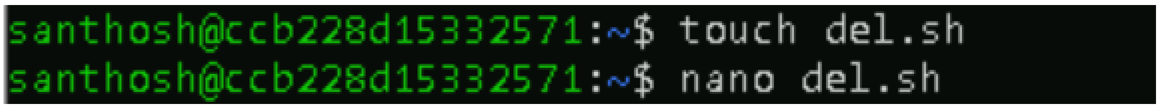


```
santhosh@ccb228d15332571:~$ ./arr8.sh
Java HTML CSS JavaScript
```

Deleting the Entire Array

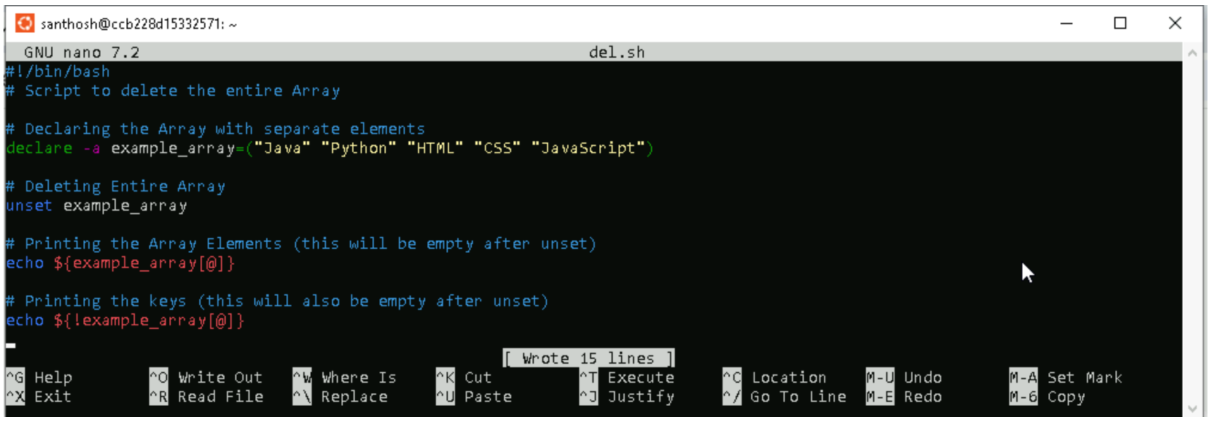
Deleting an entire array is a very simple task. It can be performed by passing the array name as an argument to the 'unset' command without specifying the index or key.

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



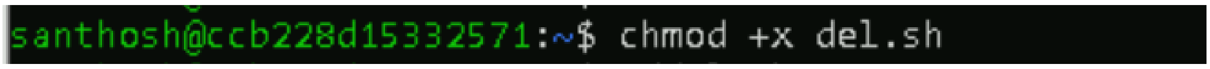
```
santhosh@ccb228d15332571:~$ touch del.sh
santhosh@ccb228d15332571:~$ nano del.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



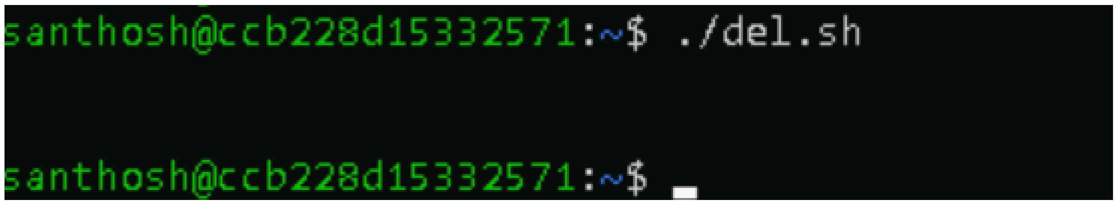
```
santhosh@ccb228d15332571: ~
  GNU nano 7.2                                  del.sh
#!/bin/bash
# Script to delete the entire Array

# Declaring the Array with separate elements
declare -a example_array=("Java" "Python" "HTML" "CSS" "JavaScript")

# Deleting Entire Array
unset example_array

# Printing the Array Elements (this will be empty after unset)
echo ${example_array[@]}

# Printing the keys (this will also be empty after unset)
echo ${!example_array[@]}

                                  [ Wrote 15 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x del.sh
```

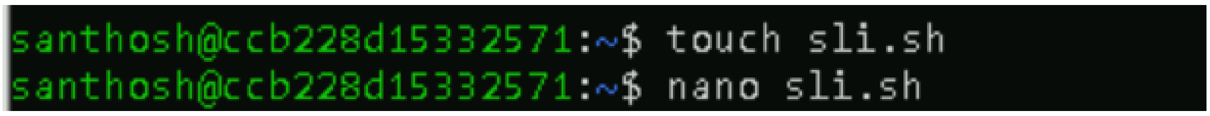
Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./del.sh


santhosh@ccb228d15332571:~$ _
```

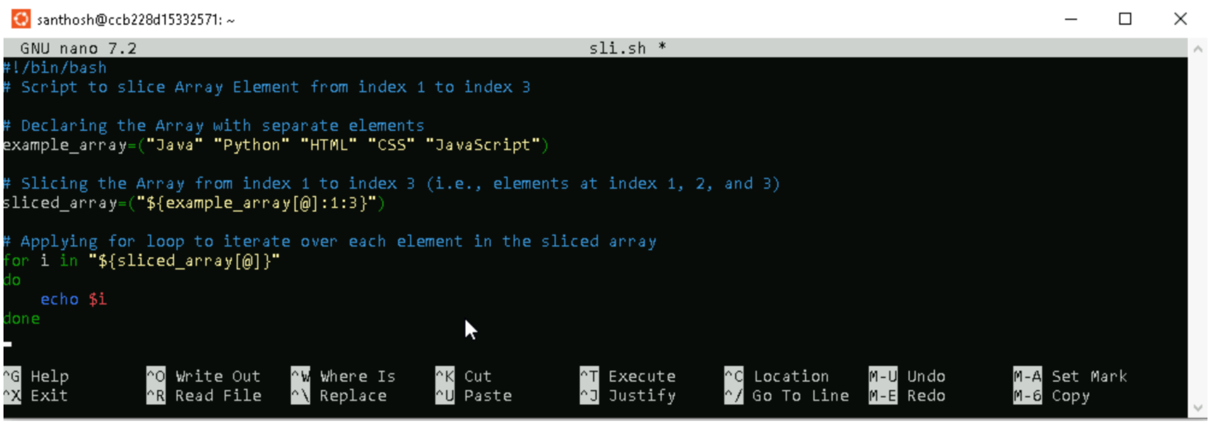
Slice Array Elements

Step 1: Creating a bash script using touch command and adding the script by editing the file using nano command.



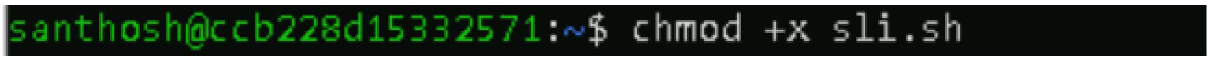
```
santhosh@ccb228d15332571:~$ touch sli.sh
santhosh@ccb228d15332571:~$ nano sli.sh
```

Step 2: Creating the script for a simple scenario to demonstrate the use of the case statement.



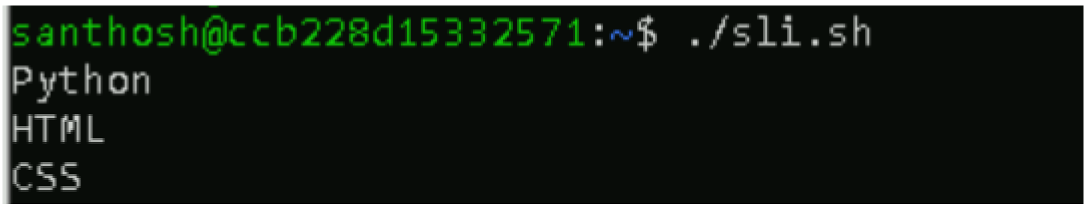
```
#!/bin/bash
# Script to slice Array Element from index 1 to index 3

# Declaring the Array with separate elements
example_array=("Java" "Python" "HTML" "CSS" "JavaScript")

# Slicing the Array from index 1 to index 3 (i.e., elements at index 1, 2, and 3)
sliced_array=("${example_array[@]:1:3}")

# Applying for loop to iterate over each element in the sliced array
for i in "${sliced_array[@]}"
do
    echo $i
done
```

Step 3: Providing the necessary permissions for the ex.sh script.



```
santhosh@ccb228d15332571:~$ chmod +x sli.sh
```

Step 4: Executing the output.



```
santhosh@ccb228d15332571:~$ ./sli.sh
Python
HTML
CSS
```