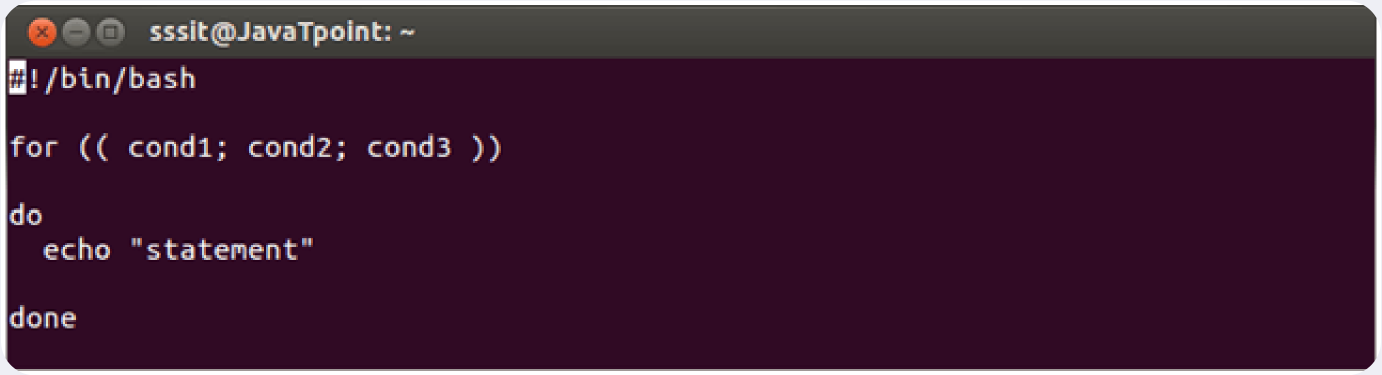
Shell Scripting for loop

The for loop moves through a specified list of values until the list is exhausted.

Syntax:

Syntax of for loop using in and list of values is shown below. This for loop contains a number of variables in the list and will execute for each item in the list. For example, if there are 10 variables in the list, then the loop will execute ten times and the value will be stored in varname.

```
for varname in value1 value2 value3 ...
do
   # Statements to be executed
done
```



```
#!/bin/bash

for (( cond1; cond2; cond3 ))

do
  echo "statement"

done
```

Made with Gamma

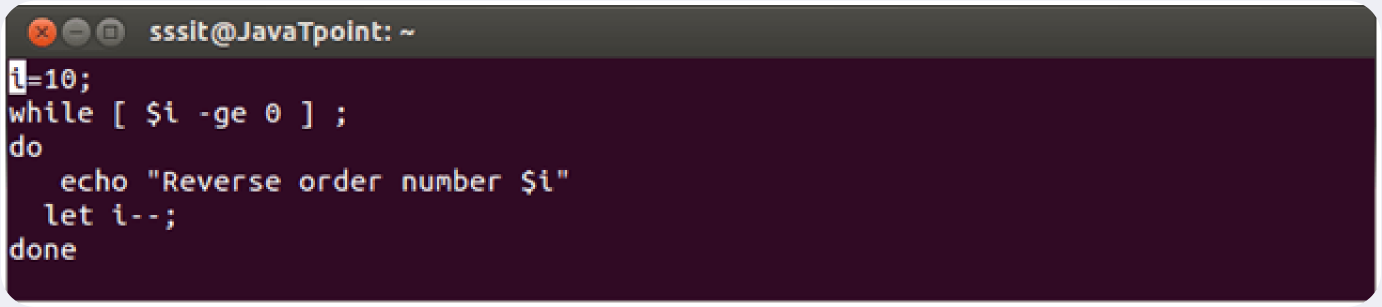
Shell Scripting while loop

Linux scripting while loop is similar to the C language while loop. The loop continues executing commands as long as the specified condition is true. Once the condition becomes false, the loop terminates.

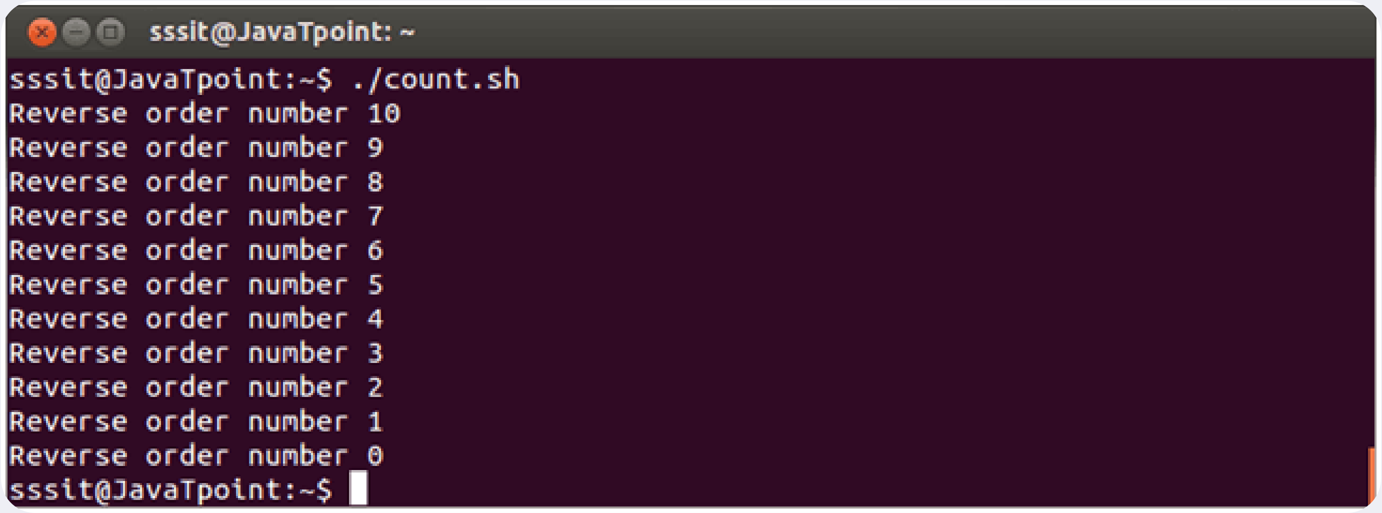
Syntax:

The syntax for the while loop in Linux scripting is as follows:

```
while condition
do
  # Commands to be executed
done
```



```
i=10;
while [ $i -ge 0 ] ;
do
   echo "Reverse order number $i"
  let i--;
done
```



```
sssit@JavaTpoint:~$ ./count.sh
Reverse order number 10
Reverse order number 9
Reverse order number 8
Reverse order number 7
Reverse order number 6
Reverse order number 5
Reverse order number 4
Reverse order number 3
Reverse order number 2
Reverse order number 1
Reverse order number 0
sssit@JavaTpoint:~$
```

Conditional Statements in Shell Script

Conditional statements are essential in Shell Scripting to control the flow of execution based on specified conditions. The five conditional statements in Bash programming are:

1 if statement

This block will be executed if a specified condition is true.

Syntax:

```
if [ expression ]
then
  statement
fi
```

2 if-else statement

If the specified condition is not true, the else part will be executed.

Syntax:

```
if [ expression ]
then
  statement1
else
  statement2
fi
```

3 if..elif..else..fi statement (Else If ladder)

Multiple conditions can be used in one if-else block using the elif keyword. If expression1 is true, it executes statement1 and statement2.

Syntax:

```
if [ expression1 ]
then
  statement1
  statement2
  .
  .
elif [ expression2 ]
then
  statement3
  statement4
  .
  .
else
  statement5
fi
```

4 if..then..else..if..then..fi..fi.. (Nested if)

Nested if-else block is used when one condition is satisfied, and it checks another condition. If expression1 is false, it processes the else part and checks expression2.

Syntax:

```
if [ expression1 ]
then
  statement1
  statement2
  .
else
  if [ expression2 ]
  then
    statement3
    .
  fi
fi
```

5 switch statement

The case statement works as a switch statement. When a specified value matches a pattern, a block of statements associated with that pattern is executed.

Syntax:

```
case variable in
  Pattern1) Statement1;;
  Pattern2) Statement2;;
  .
  .
esac
```

Write a Shell script to list all of the directory files in a directory.

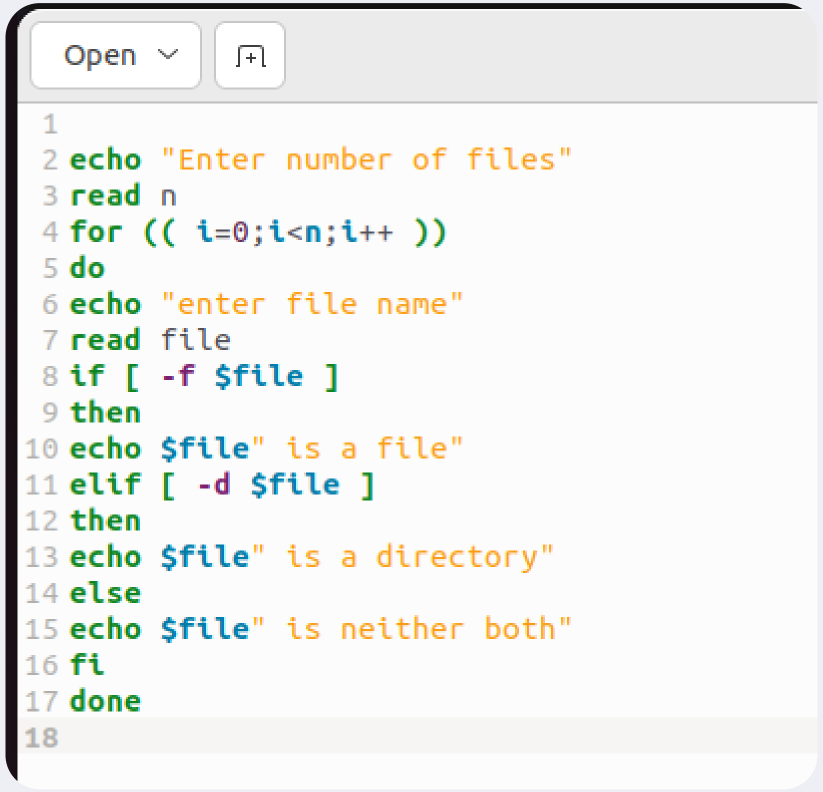
AIM:

Write a Shell script to list all of the directory files in a directory.

Program:

```
# !/bin/bash
echo "enter directory name"
read dir
if[ -d $dir]
then
echo "list of files in the directory"
ls -l $dir|egrep ‘^d’
else
echo "enter proper directory name"
fi
```

Write a shell script that receives any number of file names as arguments check if every argument supplied is a file or a directory and reports accordingly. When ever the argument is a file or directory



```

echo "Enter number of files"
read n
for (( i=0;i<n;i++ ))
do
echo "enter file name"
read file
if [ -f $file ]
then
echo $file" is a file"
elif [ -d $file ]
then
echo $file" is a directory"
else
echo $file" is neither both"
fi
done

```

Made with Gamma

Check File or Directory

Linux shell scripts give you the ability to check whether user inputs are files or directories. We can take advantage of operators -f and -d to easily verify the type of input. Use the following syntax for your scripts:

Command	Description
if[-f $file]	This tests whether $filename is a regular file.
elif [-d $file]	This tests whether $filename is a directory file.
else	This command is executed only if the two previous tests failed.