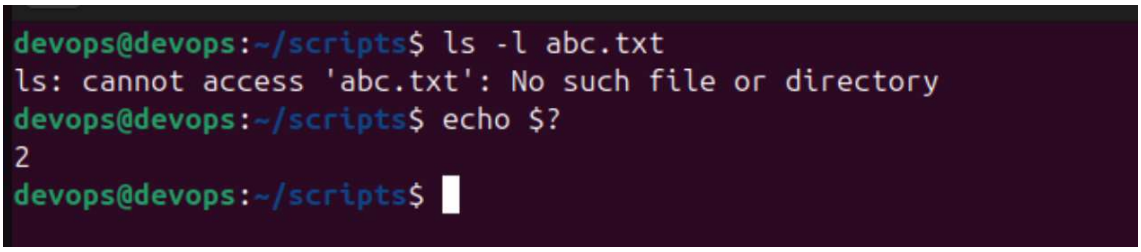
What is exit code in Linux?

- An exit code, also known as an exit status, in Linux is a numerical value returned by a command or program to indicate its execution result.
- This code provides information on whether the command completed successfully or encountered errors.
- In Linux, the exit codes range from 0 to 255, with 0 representing success, and 1-255 indicating various failure conditions.

The **$? Is a special shell variable** that stores the exit status of the most recently executed command or script.

Example: Assume the file is not present.

In shell, type:



2 – means last command was not successful.

Practice 1: Try with some command and check $? Value for success.

Output:

```
devops@devops:~/scripts$ ./exittest.sh
total 56
-rwxr--r-- 1 devops devops  478 Jun 17 14:05 arthemtic.sh
drwxrwxr-x 2 devops devops 4096 Jun 18 14:58 backup
-rwx------ 1 devops devops  109 Jun 18 15:37 backup.sh
-rwx------ 1 devops devops  333 Jun 18 14:05 comparenumber.sh
-rwx------ 1 devops devops  141 Jun 18 17:21 exittest.sh
-rwx------ 1 devops devops  190 Jun 16 10:02 hello_world.sh
-rwx------ 1 devops devops  115 Jun 18 09:52 ifelif.sh
-rwx------ 1 devops devops  237 Jun 18 11:26 ifelseif.sh
-rwx------ 1 devops devops  134 Jun 17 15:09 ifelse.sh
-rwx------ 1 devops devops  727 Jun 17 14:35 roperator.sh
-rwxr-xr-x 1 devops devops  181 Jun 16 09:41 susbs.sh
-rwx------ 1 devops devops  239 Jun 18 17:13 testexit.sh
-rwxrwx--- 1 devops devops   46 Jun 15 05:31 test.sh
-rwxr-xr-x 1 devops devops  163 Jun 17 11:31 testvar.sh
command suceesful
```

Practice 2: Try with some command and check $? exit code

```
# ! /usr/bin/bash

ls
exit_code=$?

if [ $exit_code -eq 0 ];then
        echo "command was successful"
else
        echo "command was unsuccessful with exit code: $exit_code"
fi
#else
#       echo "cat commnd was not run sucessful"
#       exit 0
#fi
```

Output:

```
devops@devops:~/scripts$ ./testexit.sh
arthemtic.sh  comparenumber.sh  ifelif.sh    roperator.sh  test.sh
backup        exittest.sh       ifelseif.sh  susbs.sh      testvar.sh
backup.sh     hello_world.sh    ifelse.sh    testexit.sh
command was successful
```

Understanding Exit Code Conventions

The Linux and Unix-like operating systems have established some common conventions for exit code usage:

- 0: Indicates success.
- 1: Indicates a general error.
- 2: Indicates an incorrect usage or invalid arguments.
- 126: Indicates a command could not be executed.
- 127: Indicates a command was not found.
- 128+n: Indicates a fatal error signal "n" (e.g., 128+9 = 137 for SIGKILL).

What is command-line arguments in Linux?

Instead of getting input from a shell program or assigning it to the program, the arguments are passed in the execution part.

Command-line arguments are passed in the positional way.

```
sh <script filename> arg1 arg2 arg3 ......
```

So, our code of execution will become

sh displayPositionalArgument.sh Welcome To GeeksForGeeks

Always the first argument starts after the <script filename>. <script filename> will be in the 0th location, We need to take positional arguments after the <script filename>. Hence in the above example

```
$1-> "Welcome"
$2-> "To"
$3-> "GeeksForFeeks"
```

Special Variable	Special Variable's details
$1 ... $n	Positional argument indicating from 1 .. n. If the argument is like 10, 11 onwards, it has to be indicated as ${10},${11
$0	This is not taken into the argument list as this indicates the "name" of the shell program. In the above example, **$0 is "displayPositionalArgument.sh"**
$@	Values of the arguments that are passed in the program. This will be much helpful if we are not sure about the number of arguments that got passed.
$#	Total number of arguments and it is a good approach for loop concepts.
$*	In order to get all the arguments as double-quoted, we can follow this way
$$	To know about the process id of the current shell
$? and $!	Exit status id and Process id of the last command

Practice 3: Display the content of a file sent as command line argument to a script. If file does not exists then display the error.

devops@devops: ~/scripts

```
# ! /usr/bin/bash

cat $1
exit_code=$?

if [ $exit_code -eq 0 ];then
        echo "command was successful"
else
        echo "command was unsuccessful with exit code: $exit_code"
fi
#else
#       echo "cat commnd was not run sucessful"
#       exit 0
#fi
~
```

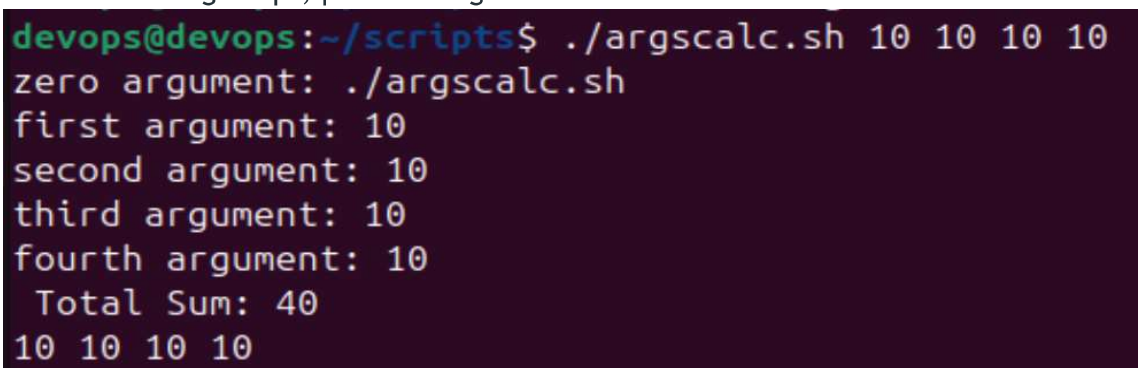
Output :

```
devops@devops:~/scripts$ ./fileargs.sh test.sh
#! /usr/bin/bash
echo "my test script"
exit 0
command was successful
devops@devops:~/scripts$ ./fileargs.sh test1.sh
cat: test1.sh: No such file or directory
command was unsuccessful with exit code: 1
```

Practice 4 : Display the arguments passed while executing the script.

Output :

While executing script, pass the arguments.



Loops

There are total 3 looping statements that can be used in bash programming.

1. For
2. While
3. Until

To alter the flow of loops, you can use 2 commands:

a. Break
a. Continue

For loop syntax:

```
#! /usr/bin/bash
for <var> in <value1 value2 ... valuen>
do
   <command 1>
   <command 2>
   <etc>
done
```

Example 1 :

```
#! /usr/bin/bash
#Start of for loop
for a in 1 2 3 4 5 6 7 8 9 10
do
   # if a is equal to 5 break the loop
   if [ $a == 5 ]
   then
     break
fi
# Print the value
echo "Iteration no $a
done
```

Output :

```
devops@devops:~/scripts$ ./forloop.sh
Iteration no 1
Iteration no 2
Iteration no 3
Iteration no 4
```

While loop syntax:

```
#! /usr/bin/bash

while <condition>
do
   <command 1>
   <command 2>
   <etc>
done
```

Example :

```
#! /usr/bin/bash
a=0
# lt is less than operator
#Iterate the loop until a less than 10
while [ $a -lt 10 ]
do
# Print the values
echo $a
# increment the value
a=`expr $a + 1`
done
```

Output :

```
devops@devops:~/scripts$ ./whileloop.sh
0
1
2
3
4
5
6
7
8
9
```

Practice Examples :

Practice 1 : To create 5 files in the current directory.

```
#! /usr/bin/bash
# Script will display 1 to 5 and will create file in eacg iteration
for i in {1..5} # iteration of i start with 1
do
        echo $i
        touch file$i.log
done

echo $i

#Loop complete
echo " Loop completed"
```

Practice 2: Print the contents of 3 files passed as command line arguments

```
#! /usr/bin/bash
#for str in $1 $2 $3
#do
#       echo $str
#       cat $str
#done
for str in $@    # $@
do
        echo $str
        echo "==========================================================="
        cat $str
done
```

While executing script, pass the arguments.

```
devops@devops:~/scripts$ ./practice2.sh testexit.sh test.sh practice1.sh
```

Practice 3 : Printing a number is odd or even.

```
#! /usr/bin/bash
# This script will check number odd or even
# Date Created : 20-Jun-25
# Author : Yogesh Devpura

echo "Enter Number to check:"
read num

remainder=$(expr $num % 2)
if [ $remainder -eq 0 ]
then
        echo "$num is an even number"
else
        echo "$num is an odd number"
fi
```

Execute the script…

```
devops@devops:~/scripts$ ./practice3.sh
Enter Number to check:
20
20 is an even number
devops@devops:~/scripts$ ./practice3.sh
Enter Number to check:
17
17 is an odd number
```

Practice 4: Check if the file name accepted by the user exists or not. If exists, print its content. Otherwise, create a new file and add content into it.

Use: -e option in if command to check if file exists in the current directory.

```
#! /usr/bin/bash
#This is script to check file exists or not

echo "Enter Filename"
read file

echo "The Filename entered by you is: $file"
# -e to check file exists or not
if [ -e $file ]
then
        echo "File exists in the directory"
        cat $file
else
        touch $file
        echo "New File create with name: $file"
fi
~
~
~
~
~
```

Execute the script…

```
devops@devops:~/scripts$ ./practice4.sh
Enter Filename
test.sh
The Filename entered by you is: test.sh
File exists in the directory
#! /usr/bin/bash
echo "my test script"
exit 0
devops@devops:~/scripts$ ./practice4.sh
Enter Filename
abc.txt
The Filename entered by you is: abc.txt
New File create with name: abc.txt
```

In this example, we created a file with **touch** command in the else block. **Add a command to create a file and add content into it in the else block.**

Practice 5 : Print numbers from 1 to 10 using while loop.

```
#! /usr/bin/bash
# sciprt to print 1 to 10 using while loop

i=1

# iterate the loop until i <= 10

while [ $i -le 10 ]
do
        echo $i
        #increment the value by i using expr command
        #i=`expr $i + 1`
        #echo $i
        i=$(expr $i + 1)
        #echo $i
done
```

Output :

```
devops@devops:~/scripts$ ./practice5.sh
1
2
3
4
5
6
7
8
9
10
```