

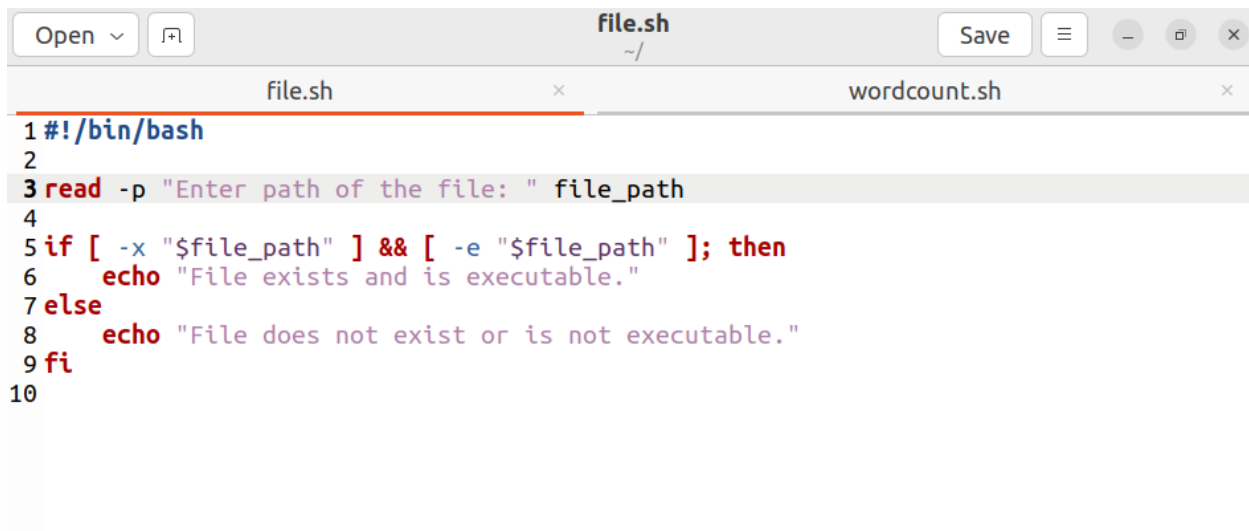
MODULE 4

ASSIGNMENT 1

Logical Operators

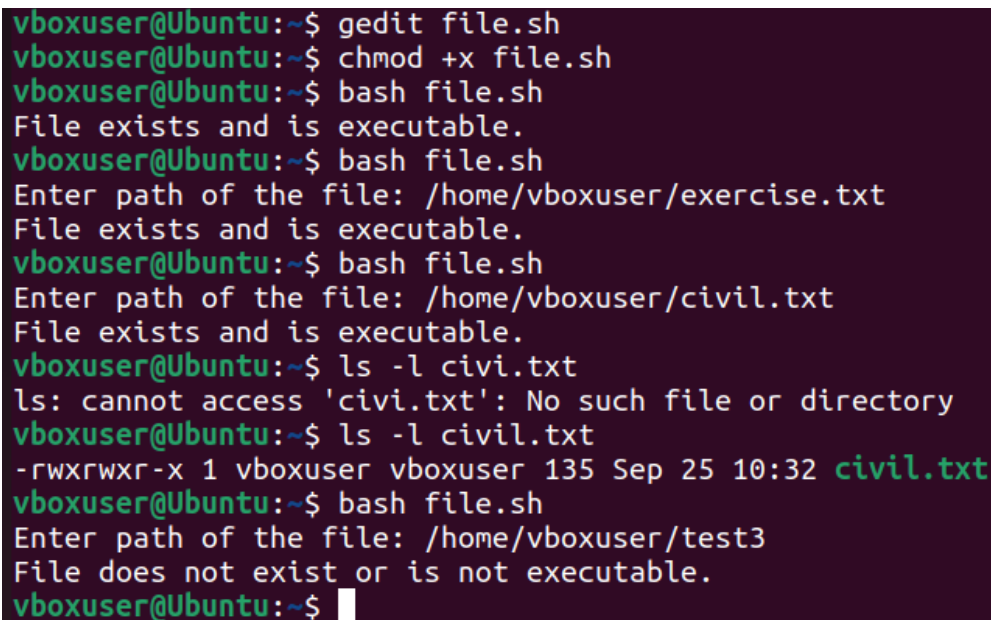
1) Check whether the file exists and is executable using logical operators.

Hint:man test



The screenshot shows a gedit editor window with two tabs: 'file.sh' and 'wordcount.sh'. The 'file.sh' tab is active and contains the following script:

```
1#!/bin/bash
2
3read -p "Enter path of the file: " file_path
4
5if [ -x "$file_path" ] && [ -e "$file_path" ]; then
6    echo "File exists and is executable."
7else
8    echo "File does not exist or is not executable."
9fi
10
```



The screenshot shows a terminal window with the following commands and output:

```
vboxuser@Ubuntu:~$ gedit file.sh
vboxuser@Ubuntu:~$ chmod +x file.sh
vboxuser@Ubuntu:~$ bash file.sh
File exists and is executable.
vboxuser@Ubuntu:~$ bash file.sh
Enter path of the file: /home/vboxuser/exercise.txt
File exists and is executable.
vboxuser@Ubuntu:~$ bash file.sh
Enter path of the file: /home/vboxuser/civil.txt
File exists and is executable.
vboxuser@Ubuntu:~$ ls -l civi.txt
ls: cannot access 'civi.txt': No such file or directory
vboxuser@Ubuntu:~$ ls -l civil.txt
-rwxrwxr-x 1 vboxuser vboxuser 135 Sep 25 10:32 civil.txt
vboxuser@Ubuntu:~$ bash file.sh
Enter path of the file: /home/vboxuser/test3
File does not exist or is not executable.
vboxuser@Ubuntu:~$
```

ASSIGNMENT 2

Arithmetic Comparison

1) Write a program to demonstrate the use of not equal to operator.

Hint: -ne

```
file.sh × wordcount.sh ×
1 #!/bin/bash
2
3 read -p "Enter filename: " y
4
5 x=`cat $y | wc -w`
6
7 if [ $x -eq 0 ]; then
8     echo "the file $y has zero words"
9 elif [ $x -ne 0 ]; then
10    echo "the file $y has $x words"
11 else
12    echo "invalid"
13 fi
14 |
```

```
vboxuser@Ubuntu:~$ gedit wordcount.sh
vboxuser@Ubuntu:~$ chmod +x wordcount.sh
vboxuser@Ubuntu:~$ bash wordcount.sh
Enter filename: array2.sh
the file array2.sh has 59 words
vboxuser@Ubuntu:~$ bash wordcount.sh
Enter filename: file.sh
the file file.sh has 38 words
vboxuser@Ubuntu:~$ bash wordcount.sh
Enter filename: names.txt
the file names.txt has 24 words
vboxuser@Ubuntu:~$ bash wordcount.sh
Enter filename: seddemo.txt
the file seddemo.txt has 25 words
vboxuser@Ubuntu:~$
```

ASSIGNMENT 3

String and File attributes

1) Explore some more attributes

-r

-X

-O

```
vboxuser@Ubuntu: ~  
vboxuser@Ubuntu:~$ gedit strcompare.sh  
vboxuser@Ubuntu:~$ chmod +x strcompare.sh  
vboxuser@Ubuntu:~$ whoami  
vboxuser  
vboxuser@Ubuntu:~$ bash strcompare.sh  
You have no permission to run strcompare.sh as non-root user.  
vboxuser@Ubuntu:~$ sudo ./strcompare.sh  
[sudo] password for vboxuser:  
welcome root!  
vboxuser@Ubuntu:~$ ls -o strcompare.sh  
-rwxrwxr-x 1 vboxuser 149 Sep 28 16:50 strcompare.sh  
vboxuser@Ubuntu:~$ ls -r strcompare.sh  
strcompare.sh  
vboxuser@Ubuntu:~$ ls -x strcompare.sh  
strcompare.sh  
vboxuser@Ubuntu:~$ touch unix.txt  
vboxuser@Ubuntu:~$ ls -o strcompare.sh  
-rwxrwxr-x 1 vboxuser 149 Sep 28 16:50 strcompare.sh  
vboxuser@Ubuntu:~$ ls -o unix.txt  
-rw-rw-r-- 1 vboxuser 0 Sep 28 17:00 unix.txt  
vboxuser@Ubuntu:~$ ls -r unix.txt  
unix.txt  
vboxuser@Ubuntu:~$ ls -w unix.txt  
ls: invalid line width: 'unix.txt'  
vboxuser@Ubuntu:~$ ls -x unix.txt  
unix.txt  
vboxuser@Ubuntu:~$ ls -s unix.txt  
0 unix.txt  
vboxuser@Ubuntu:~$
```

ASSIGNMENT 4

Conditional Loops

- 1) Find the sum of first n prime numbers.


```
Open  +  prime.sh  Save  -  ×
1 #!/bin/bash
2
3 read -p "Enter the limit to print prime numbers: " n
4
5 sum=0
6 count=0
7 number=2
8
9 while [ $count -lt $n ]; do
10     is_prime=true
11
12     for ((i = 2; i * i <= number; i++)); do
13         if [ $((number % i)) -eq 0 ]; then
14             is_prime=false
15             break
16         fi
17     done
18
19     if $is_prime; then
20         sum=$((sum + number))
21         count=$((count + 1))
22     fi
23
24     number=$((number + 1))
25 done
26
27 echo "The sum of the first $n prime numbers is: $sum"
28 |
```

```
vboxuser@Ubuntu:~$ bash prime.sh
Enter the limit to print prime numbers: 4
The sum of the first 4 prime numbers is: 17
vboxuser@Ubuntu:~$ bash prime.sh
Enter the limit to print prime numbers: 5
The sum of the first 5 prime numbers is: 28
```

ASSIGNMENT 5

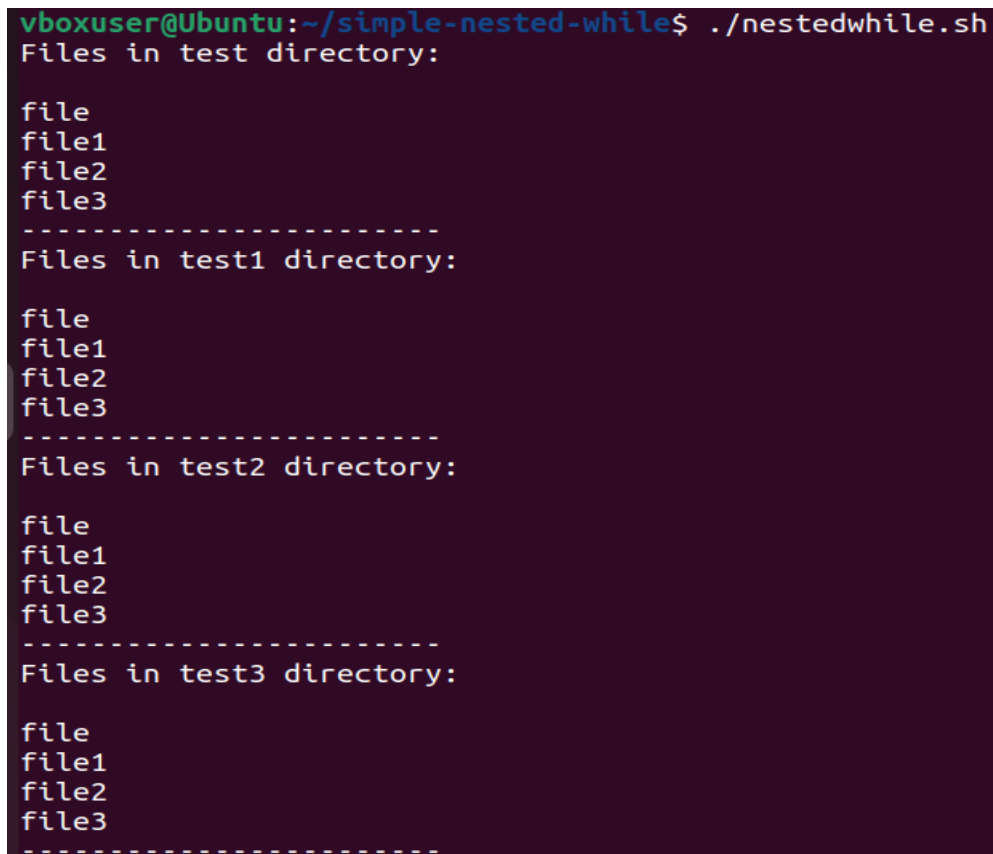
More on Loops

- 1) Retype nested-for.sh bash script using nested while loop
- 2) Save your program with the name: nested-while.sh



The screenshot shows a code editor window with the title 'nestedwhile.sh' and a file icon. The editor contains the following bash script:

```
1 for dir in test*; do
2     echo "Files in $dir directory:"
3     echo ""
4     for file in $(ls -1 $dir); do
5         echo $file
6     done
7     echo "-----"
8 done
```



The screenshot shows a terminal window with the prompt 'vboxuser@Ubuntu:~/simple-nested-while\$' and the command './nestedwhile.sh'. The output of the script is as follows:

```
vboxuser@Ubuntu:~/simple-nested-while$ ./nestedwhile.sh
Files in test directory:

file
file1
file2
file3
-----
Files in test1 directory:

file
file1
file2
file3
-----
Files in test2 directory:

file
file1
file2
file3
-----
Files in test3 directory:

file
file1
file2
file3
-----
```

ASSIGNMENT 6

1) Write a menu driven program for mathematical calculation

- It should take user inputs a and b
- It should ask for mathematical operator (+, -, / and *).
- Do the calculation
- Print the output

```
calculator.sh
~/
Save
prime.sh calculator.sh
1 #!/bin/bash
2
3 while true; do
4     echo "Mathematical Calculator Menu:"
5     echo "1. Addition (+)"
6     echo "2. Subtraction (-)"
7     echo "3. Multiplication (*)"
8     echo "4. Division (/)"
9     echo "5. Exit"
10    read -p "Choose an operation (1/2/3/4/5): " choice
11
12    case $choice in
13        1)
14            read -p "Enter the first number (a): " a
15            read -p "Enter the second number (b): " b
16            result=$((a + b))
17            echo "Result: $a + $b = $result"
18            ;;
19        2)
20            read -p "Enter the first number (a): " a
21            read -p "Enter the second number (b): " b
22            result=$((a - b))
23            echo "Result: $a - $b = $result"
24            ;;
25        3)
26            read -p "Enter the first number (a): " a
27            read -p "Enter the second number (b): " b
28            result=$((a * b))
29            echo "Result: $a * $b = $result"
30            ;;
31        4)
32            read -p "Enter the first number (a): " a
33            read -p "Enter the second number (b): " b
34            if [ "$b" -eq 0 ]; then
35                echo "Error: Division by zero is not allowed."
36            else
37                result=$(echo "scale=2; $a / $b" | bc)
38                echo "Result: $a / $b = $result"
39            fi
40            ;;
41        5)
42            echo "Exiting the calculator."
43            exit 0
44            ;;
45        *)
46            echo "Invalid choice. Please select a valid option (1/2/3/4/5)."
47            ;;
48    esac
49 done
50
```

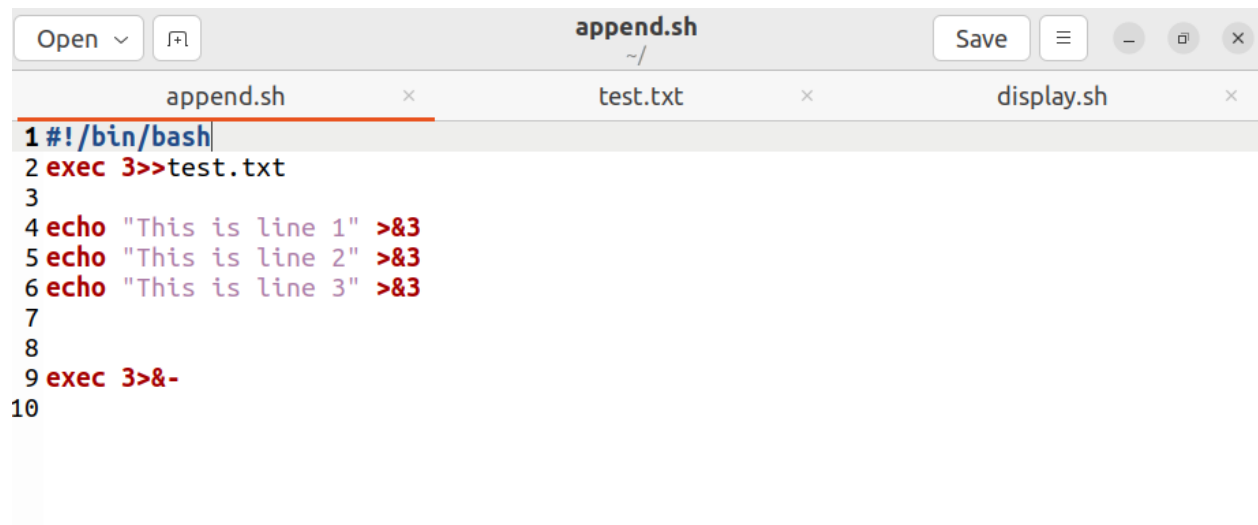
```
vboxuser@Ubuntu:~$ gedit calculator.sh
vboxuser@Ubuntu:~$ chmod +x calculator.sh
vboxuser@Ubuntu:~$ ./calculator.sh
Mathematical Calculator Menu:
1. Addition (+)
2. Subtraction (-)
3. Multiplication (*)
4. Division (/)
5. Exit
Choose an operation (1/2/3/4/5): 1
Enter the first number (a): 2
Enter the second number (b): 5
Result: 2 + 5 = 7
Mathematical Calculator Menu:
1. Addition (+)
2. Subtraction (-)
3. Multiplication (*)
4. Division (/)
5. Exit
Choose an operation (1/2/3/4/5): 2
Enter the first number (a): 45
Enter the second number (b): 24
Result: 45 - 24 = 21
Mathematical Calculator Menu:
1. Addition (+)
2. Subtraction (-)
3. Multiplication (*)
4. Division (/)
5. Exit
Choose an operation (1/2/3/4/5): 3
Enter the first number (a): 4
Enter the second number (b): 5
Result: 4 * 5 = 20
```

```
Mathematical Calculator Menu:
1. Addition (+)
2. Subtraction (-)
3. Multiplication (*)
4. Division (/)
5. Exit
Choose an operation (1/2/3/4/5): 5
Exiting the calculator.
```

ASSIGNMENT 7

Using File Descriptors

- 1) Try to append few lines to a file test.txt using file descriptor.
- 2) Display the content of the file using file descriptor.



A terminal window titled 'append.sh' with a path of '~/'. The window contains the following commands and their outputs:

```
1 #!/bin/bash
2 exec 3>>test.txt
3
4 echo "This is line 1" >&3
5 echo "This is line 2" >&3
6 echo "This is line 3" >&3
7
8
9 exec 3>&-
10
```



A terminal window titled 'test.txt' with a path of '~/'. The window displays the contents of the file test.txt, which were appended in the previous step:

```
1 This is line 1
2 This is line 2
3 This is line 3
```



```
append.sh × test.txt × display.sh ×
1#!/bin/bash
2exec 4<test.txt
3
4
5while IFS= read -r line; do
6    echo "$line"
7done <&4
8
9
10exec 4<&-
11
```

```
vboxuser@Ubuntu:~$ gedit append.sh
vboxuser@Ubuntu:~$ chmod +x append.sh
vboxuser@Ubuntu:~$ ./append.sh
vboxuser@Ubuntu:~$ gedit test.txt
vboxuser@Ubuntu:~$ gedit display.sh
vboxuser@Ubuntu:~$ chmod +x display.sh
vboxuser@Ubuntu:~$ ./display.sh
This is line 1
This is line 2
This is line 3
vboxuser@Ubuntu:~$
```

ASSIGNMENT 8

Basics of functions

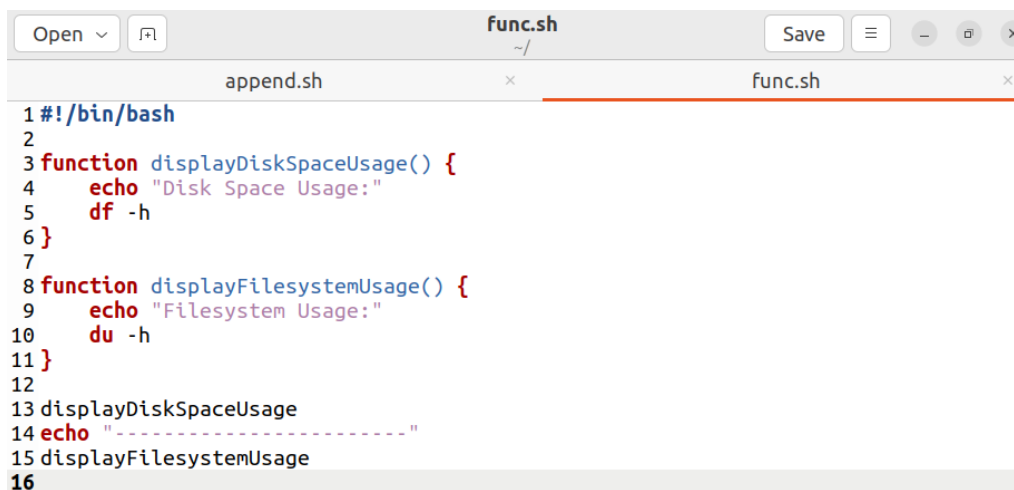
1) Write a program with two functions:

- The first function should display disk space usage in human readable form.

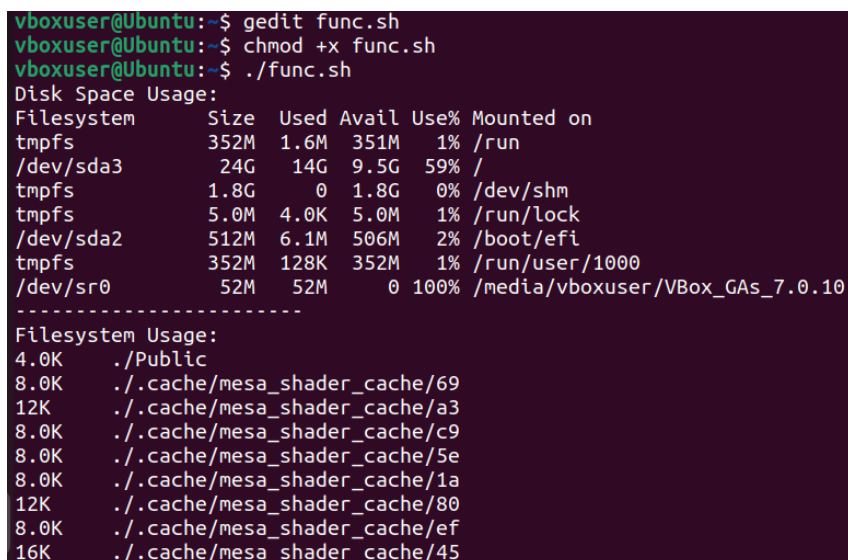
(Hint: `df -h`)

- The second function should display filesystem usage in human readable form.

(Hint: `du -h`)



```
1#!/bin/bash
2
3function displayDiskSpaceUsage() {
4    echo "Disk Space Usage:"
5    df -h
6}
7
8function displayFilesystemUsage() {
9    echo "Filesystem Usage:"
10   du -h
11}
12
13displayDiskSpaceUsage
14echo "-----"
15displayFilesystemUsage
16
```

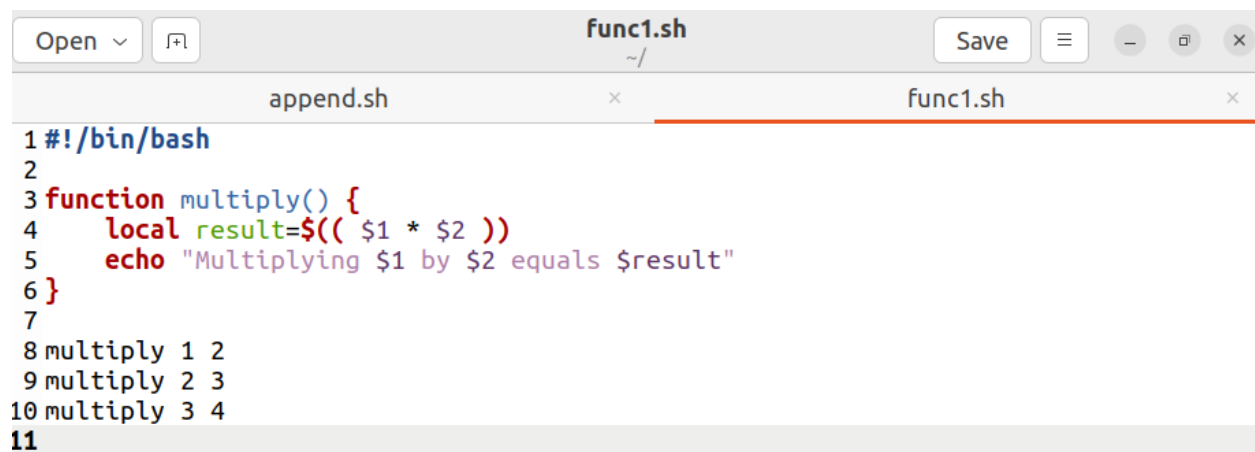


```
vboxuser@Ubuntu:~$ gedit func.sh
vboxuser@Ubuntu:~$ chmod +x func.sh
vboxuser@Ubuntu:~$ ./func.sh
Disk Space Usage:
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           352M   1.6M   351M   1% /run
/dev/sda3       24G    14G   9.5G   59% /
tmpfs           1.8G    0   1.8G    0% /dev/shm
tmpfs           5.0M   4.0K   5.0M   1% /run/lock
/dev/sda2       512M   6.1M   506M   2% /boot/efi
tmpfs           352M   128K   352M   1% /run/user/1000
/dev/sr0        52M    52M    0  100% /media/vboxuser/VBox_GAs_7.0.10
-----
Filesystem Usage:
4.0K  ./Public
8.0K  ./cache/mesa_shader_cache/69
12K   ./cache/mesa_shader_cache/a3
8.0K  ./cache/mesa_shader_cache/c9
8.0K  ./cache/mesa_shader_cache/5e
8.0K  ./cache/mesa_shader_cache/1a
12K   ./cache/mesa_shader_cache/80
8.0K  ./cache/mesa_shader_cache/ef
16K   ./cache/mesa_shader_cache/45
```

ASSIGNMENT 9

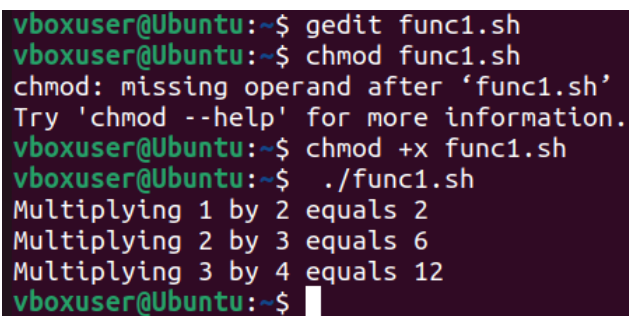
More on functions

- 1) Write a program,
 - a. where the function accepts two arguments.
 - b. The function should multiply the two arguments.
 - c. Make 3 function calls with arguments - (1, 2), (2, 3) and (3, 4)



The screenshot shows a gedit editor window with the title bar 'func1.sh' and a path indicator '~/'. The window contains the following script:

```
1#!/bin/bash
2
3function multiply() {
4    local result=$(( $1 * $2 ))
5    echo "Multiplying $1 by $2 equals $result"
6}
7
8multiply 1 2
9multiply 2 3
10multiply 3 4
11
```



The screenshot shows a terminal window with the following commands and output:

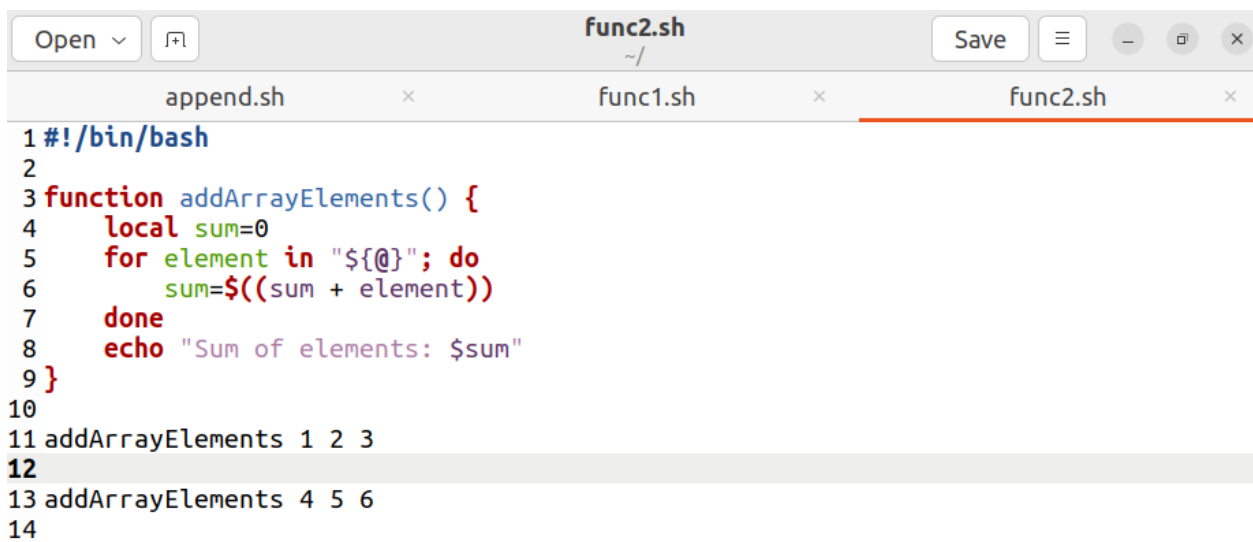
```
vboxuser@Ubuntu:~$ gedit func1.sh
vboxuser@Ubuntu:~$ chmod func1.sh
chmod: missing operand after 'func1.sh'
Try 'chmod --help' for more information.
vboxuser@Ubuntu:~$ chmod +x func1.sh
vboxuser@Ubuntu:~$ ./func1.sh
Multiplying 1 by 2 equals 2
Multiplying 2 by 3 equals 6
Multiplying 3 by 4 equals 12
vboxuser@Ubuntu:~$
```

ASSIGNMENT 10

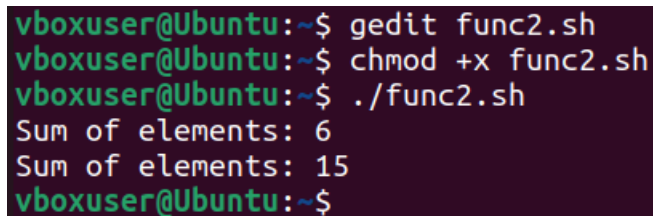
Arrays and functions

1) Write a program,

- a. Where a function adds all the elements in an array.
- b. The function should display the sum of elements.
- c. Make 2 function calls with array elements- (1, 2, 3) and (4, 5, 6).



```
1#!/bin/bash
2
3function addArrayElements() {
4    local sum=0
5    for element in "${@}"; do
6        sum=$((sum + element))
7    done
8    echo "Sum of elements: $sum"
9}
10
11addArrayElements 1 2 3
12
13addArrayElements 4 5 6
14
```



```
vboxuser@Ubuntu:~$ gedit func2.sh
vboxuser@Ubuntu:~$ chmod +x func2.sh
vboxuser@Ubuntu:~$ ./func2.sh
Sum of elements: 6
Sum of elements: 15
vboxuser@Ubuntu:~$
```