Практическая работа №8. Создание сценариев

1. В вашем рабочем каталоге создайте папку “MyScripts” с использованием команды mkdir.

   ```
   mkdir MyScripts
   ```

2. Перейдите в каталог “MyScripts”, используя команду cd.

   ```
   cd MyScripts
   ```

3. Создайте в каталоге “MyScripts” файл “MyScript1.sh” используя команду nano.

   ```
   nano MyScript1.sh
   ```

4. В открывшемся файле напишите ваш первый bash-скрипт.

   ```
   #!/bin/bash

   echo "This is Message from Script"
   echo "Hello"
   ```

5. Сохраните и закройте файл вашего первого bash-скрипта.

6. Запустите ваш bash-скрипт:

   ```
   ./MyScript1.sh
   ```

7. Оцените результат выполнения команды.

   ```
   gordeevas@MRS4G0-PC:~/MyScripts$ ./MyScript1.sh
   -bash: ./MyScript1.sh: Permission denied
   ```

8. Запустите ваш bash-скрипт, используя команду bash.

   ```
   bash MyScript1.sh
   ```

   ```
   gordeevas@MRS4G0-PC:~/MyScripts$ bash MyScript1.sh
   This is Message from Script
   Hello
   ```

9. Сравните результат выполнения команды с предыдущем вызовом (см. п.6).

   В пункте 6 мы запустили bash-скрипт, указав в качестве команды его имя, но не установили для него права на выполнение. Поэтому, в результате, мы получили ошибку. В пункте 8 мы запустили bash-скрипт, указав в качестве команды bash, а затем имя bash-скрипта.

10. Создайте в каталоге “MyScripts” файл “MyScript2.sh” используя команду nano.

```
nano MyScript2.sh
```

11. В теле нового bash-скрипта запишите следующий код.

```
#!/bin/bash

echo "Let's show files in this folder"

ls -l

echo "Done"
```

12. Сохраните и закройте ваш новый скрипт. Вызовите его выполнение, используя команду bash.

```
bash MyScript2.sh
```

13. В результате, у вас должен появиться список файлов в каталоге “MyScripts” с назначенными правами доступа.

```
gordeevas@MRS4G0-PC:~/MyScripts$ bash MyScript2.sh
Let's show files in this folder
total 8
-rw-r--r-- 1 gordeevas gordeevas 61 Feb 13 23:08 MyScript1.sh
-rw-r--r-- 1 gordeevas gordeevas 72 Feb 13 23:12 MyScript2.sh
Done
```

14. Измените права доступа к этим двум файлам, используя команду chmod.

```
chmod a+x MyScript1.sh MyScript2.sh
```

15. Запустите bash-скрипт “MyScript2.sh”.

```
./MyScript2.sh
```

16. Оцените результат выполнения команды из п.14.

    Т.к. мы установили права на выполнение для bash-скриптов, то теперь мы можем запускать их, указывая в качестве команды их имя.

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./MyScript2.sh
Let's show files in this folder
total 8
-rwxr-xr-x 1 gordeevas gordeevas 61 Feb 13 23:08 MyScript1.sh
-rwxr-xr-x 1 gordeevas gordeevas 72 Feb 13 23:12 MyScript2.sh
Done
```

17. В bash-скрипте “MyScript3.sh” введите указанный ниже код. Затем выполните этот bash-скрипт.

```
nano MyScript3.sh
```

```
#!/bin/bash

myOS=`uname -a`

echo "My Operating System"
echo "$myOS"
```

```
chmod a+x MyScript3.sh
./MyScript3.sh
```

18. Оцените результат выполнения bash-скрипта.

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./MyScript3.sh
My Operating System
Linux MRS4G0-PC 5.15.79.1-microsoft-standard-WSL2 #1 SMP Wed Nov
23 01:01:46 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
```

19. Создайте bash-скрипт “MyScript4.sh”. В его теле укажите следующий код.

```
nano MyScript4.sh
```

```
#!/bin/bash

echo "This script name is $0"

echo "Hello, $1"
```

```
chmod a+x MyScript4.sh
```

20. Выполните bash-скрипт, указав входное значение свое имя.

```
./MyScript4.sh Alexander
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./MyScript4.sh Alexander
This script name is ./MyScript4.sh
Hello, Alexander
```

21. Модифицируйте код, как показано ниже.

```
#!/bin/bash

echo "This script name is $0"

echo "Hello, $1"
echo "Hi, $2"
```

22. Запустите bash-скрипт, указав уже два входных значения. Например, так.

```
./MyScript4.sh Alexander Ivan
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./MyScript4.sh Alexander Ivan
This script name is ./MyScript4.sh
Hello, Alexander
Hi, Ivan
```

23. Создайте bash-скрипт “MyScript5.sh”. В его теле укажите следующий код.

```
nano MyScript5.sh
```

```
#!/bin/bash

Num1=123
Num2=456
Num3=789

Summa=$((Num1+Num2+Num3))

echo "$Num1 + $Num2 + $Num3 = $Summa"
```

```
chmod a+x MyScript5.sh
```

24. Выполните bash-скрипт. Оцените результат.

```
./MyScript5.sh
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./MyScript5.sh
123 + 456 + 789 = 1368
```

25. Создайте bash-скрипт “MyScript6.sh”. В его теле укажите следующий код.

```
nano MyScript6.sh
```

```
#!/bin/bash

myHost=`hostname`
myGTW='8.8.8.8'

ping -c 4 $myGTW
traceroute $myGTW

echo -n "This is done..."
echo "Really done"
```

```
chmod a+x MyScript6.sh
```

26. Выполните bash-скрипт и оцените результат.

```
./MyScript6.sh
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./MyScript6.sh
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=102 time=48.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=102 time=48.9 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=102 time=47.6 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=102 time=48.2 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 47.618/48.249/48.912/0.460 ms
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  MRS4G0-PC.mshome.net (172.29.128.1)  0.539 ms  0.514 ms  0.500 ms
 2  192.168.1.1 (192.168.1.1)  6.112 ms  6.099 ms  6.188 ms
 3  92.43.187.1 (92.43.187.1)  6.423 ms  6.978 ms  6.308 ms
 4  10.221.128.33 (10.221.128.33)  10.082 ms  9.949 ms  9.932 ms
 5  kir-cr01-ae5.705.chel.mts-internet.net (212.188.22.97)  8.554 ms  8.542 ms  8.521 ms
 6  psshag-cr01-ae1.74.chel.mts-internet.net (212.188.0.61)  38.410 ms  34.842 ms  34.819 ms
 7  che-cr02-ae10.63.sam.mts-internet.net (212.188.42.129)  34.808 ms  35.794 ms  35.775 ms
 8  * * *
 9  m9-cr04-be7.77.msk.mts-internet.net (195.34.53.201)  35.718 ms  35.667 ms  35.653 ms
10  209.85.149.166 (209.85.149.166)  36.220 ms  36.207 ms  36.194 ms
```

27. Создайте в каталоге “MyScripts” файл “Script1.sh”. В теле bash-скрипта запишите указанный код.

```
nano Script1.sh
```

```
#!/bin/bash

if [ "$1" -lt 5 ]; then
    echo "$1 < 5"
elif [ "$1" -ge 10 ]; then
    echo "$1 >= 10"
else
    echo "5 <= $1 < 10"
fi
```

```
chmod a+x Script1.sh
```

Сохраните bash-скрипт и примените его с одним целочисленным параметром несколько раз (например, с параметром 2, 7, 12).

```
./Script1.sh 2
./Script1.sh 7
./Script1.sh 12
```

Изучите результат для того, чтобы понять, как работает код bash-скрипта.

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script1.sh 2
2 < 5
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script1.sh 7
5 <= 7 < 10
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script1.sh 12
12 >= 10
```

Создайте новый или модифицируйте этот bash-скрипт, используя следующие операции сравнения: -eq, -ne, -gt, -ge, -lt, -le.

```
#!/bin/bash

if [ "$1" -lt 5 ]; then
    echo "$1 < 5"
elif [ "$1" -ge 10 ]; then
    echo "$1 >= 10"
elif [ "$1" -lt 10 ]; then
    echo "5 <= $1 < 10"
fi
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script1.sh 12
12 >= 10
```

28. Создайте в каталоге “MyScripts” файл “Script2.sh”. В теле bash-скрипта запишите указанный код.

```
nano Script2.sh
```

```
#!/bin/bash

if [ "$1" == "$2" ]; then
    echo "$1 equals $2"
else
    echo "$1 is not equals to $2"
fi

if [ "$1" \> "$2" ]; then
    echo "ASCII: $1 > $2"
elif [ "$1" = "$2" ]; then
    echo "ASCII: $1 = $2"
else
    echo "ASCII: $1 < $2"
fi

if [ -z "$1" ]; then
    echo "variable 1 is empty"
else
    echo "variable 1 is not empty"
fi

if [ -n "$2" ]; then
    echo "variable 2 is not empty"
else
    echo "variable 2 is empty"
fi
```

```
chmod a+x Script2.sh
```

Сохраните bash-скрипт и примените его с двумя строковыми параметрами несколько раз (например, с параметрами “Str” и “str”, “str” и “Str”, “STR1” и “STR”). Примените bash-скрипт с

одним параметром, например, с “STR”. Изучите результат для того, чтобы понять, как работает код bash-скрипта.

```
./Script2.sh Str str
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script2.sh Str str
Str is not equals to str
ASCII: Str < str
variable 1 is not empty
variable 2 is not empty
```

```
./Script2.sh str Str
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script2.sh str Str
str is not equals to Str
ASCII: str > Str
variable 1 is not empty
variable 2 is not empty
```

```
./Script2.sh STR1 STR
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script2.sh STR1 STR
STR1 is not equals to STR
ASCII: STR1 > STR
variable 1 is not empty
variable 2 is not empty
```

```
./Script2.sh STR
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script2.sh STR
STR is not equals to
ASCII: STR >
variable 1 is not empty
variable 2 is empty
```

29. Создайте в каталоге “MyScripts” файл “Script3.sh”. В теле bash-скрипта запишите указанный код.

```
nano Script3.sh
```

```
#!/bin/bash

echo "Starting CASE selection..."

read -p "Enter something: " x
case $x in
    [1-5])
            echo "One - Five"
        ;;
    [6-9])
            echo "Six - Nine"
        ;;
    "Str")
            echo "Stroka"
```

```
        ;;
    *)
            echo "Parameter Unknown, sorry!"
        ;;
esac
```

```
chmod a+x Script3.sh
```

Сохраните bash-скрипт и примените его. В качестве переменной после начала работы скрипта введите один параметр. Выполните процедуру несколько раз (например, с параметром 2, 7, “Str”, “LabuLabuDapDap”). Изучите результат для того, чтобы понять, как работает код bash-скрипта.

```
./Script3.sh 2
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script3.sh
Starting CASE selection...
Enter something: 2
One - Five
```

```
./Script3.sh 7
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script3.sh
Starting CASE selection...
Enter something: 7
Six - Nine
```

```
./Script3.sh Str
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script3.sh
Starting CASE selection...
Enter something: Str
Stroka
```

```
./Script3.sh LabuLabuDapDap
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script3.sh
Starting CASE selection...
Enter something: LabuLabuDapDap
Parameter Unknown, sorry!
```

30. Создайте в каталоге “MyScripts” файл “Script4.sh”. В теле bash-скрипта запишите указанный код.

```
nano Script4.sh
```

```
#!/bin/bash

echo -n "START = "
read START

echo -n "END = "
read END
```

```
#echo "$START $END"

if [ $START -lt $END ]; then
#   echo "START > END"
    while [ $START -le $END ]; do
        echo -n "$START "
        START=$(($START+1))
#       let START=START+1
#       let START+=1
    done
elif [ $START -gt $END ]; then
#   echo "START < END"
    while [ $START -ge $END ]; do
        echo -n "$START "
        START=$(($START-1))
    done
else
    echo "START = END, or ERROR"
fi

echo ""
```

```
chmod a+x Script4.sh
```

Сохраните bash-скрипт и примените его несколько раз. В первый раз – первый аргумент должен быть больше второго (например, 12 и 22). Во второй раз – второй аргумент больше первого (например, 16 и 4). В третий раз – первый и второй аргументы равны. В четвертый раз – вместо одного из аргументов введите символы (например, 12 и “STR”, или “STOP” и 1). Изучите результат для того, чтобы понять, как работает код bash-скрипта.

```
./Script4.sh 12 22
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script4.sh
START = 12
END = 22
12 13 14 15 16 17 18 19 20 21 22
```

```
./Script4.sh 16 4
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script4.sh
START = 16
END = 4
16 15 14 13 12 11 10 9 8 7 6 5 4
```

```
./Script4.sh 12 12
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script4.sh
START = 12
END = 12
START = END, or ERROR
```

```
./Script4.sh 12 STR
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script4.sh
START = 12
END = STR
./Script4.sh: line 11: [: STR: integer expression expected
./Script4.sh: line 19: [: STR: integer expression expected
START = END, or ERROR
```

```
./Script4.sh STOP 1
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script4.sh
START = STOP
END = 1
./Script4.sh: line 11: [: STOP: integer expression expected
./Script4.sh: line 19: [: STOP: integer expression expected
START = END, or ERROR
```

31. В окне терминала создайте несколько текстовых файлов (не меньше 3х), как показано в примере.

```
echo "Hello World from file1.txt" >> file1.txt
echo "This is text in file file2.txt" >> file2.txt
echo "Bye World from file3.txt" >> file3.txt
```

Просмотрите список созданных вами текстовых файлов.

```
ls *.txt
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ echo "Hello World from file1.txt" >> file1.txt
gordeevas@MRS4G0-PC:~/MyScripts$ echo "This is text in file file2.txt" >> file2.txt
gordeevas@MRS4G0-PC:~/MyScripts$ echo "Bye World from file3.txt" >> file3.txt
gordeevas@MRS4G0-PC:~/MyScripts$ ls *.txt
file1.txt  file2.txt  file3.txt
```

Создайте в каталоге “MyScripts” файл “Script5.sh”. В теле bash-скрипта запишите указанный код.

```
nano Script5.sh
```

```
#!/bin/bash

for myFile in 'ls *.txt'; do
    cat $myFile
done
```

```
chmod a+x Script5.sh
```

Сохраните bash-скрипт и примените его. Изучите результат для того, чтобы понять, как работает код bash-скрипта.

```
./Script5.sh
```

```
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script5.sh
cat: ls: No such file or directory
Hello World from file1.txt
This is text in file file2.txt
Bye World from file3.txt
```

Модифицируйте код bash-скрипта.

```
#!/bin/bash

for myfile in 'ls *.txt'; do
    cat $myfile
done

echo -n "" > myfile.txt
for x in {1..20}; do
    echo "X = $x" >> myfile.txt
done
```

Сохраните bash-скрипт и примените его. Изучите результат для того, чтобы понять, как работает код bash-скрипта.

./Script5.sh

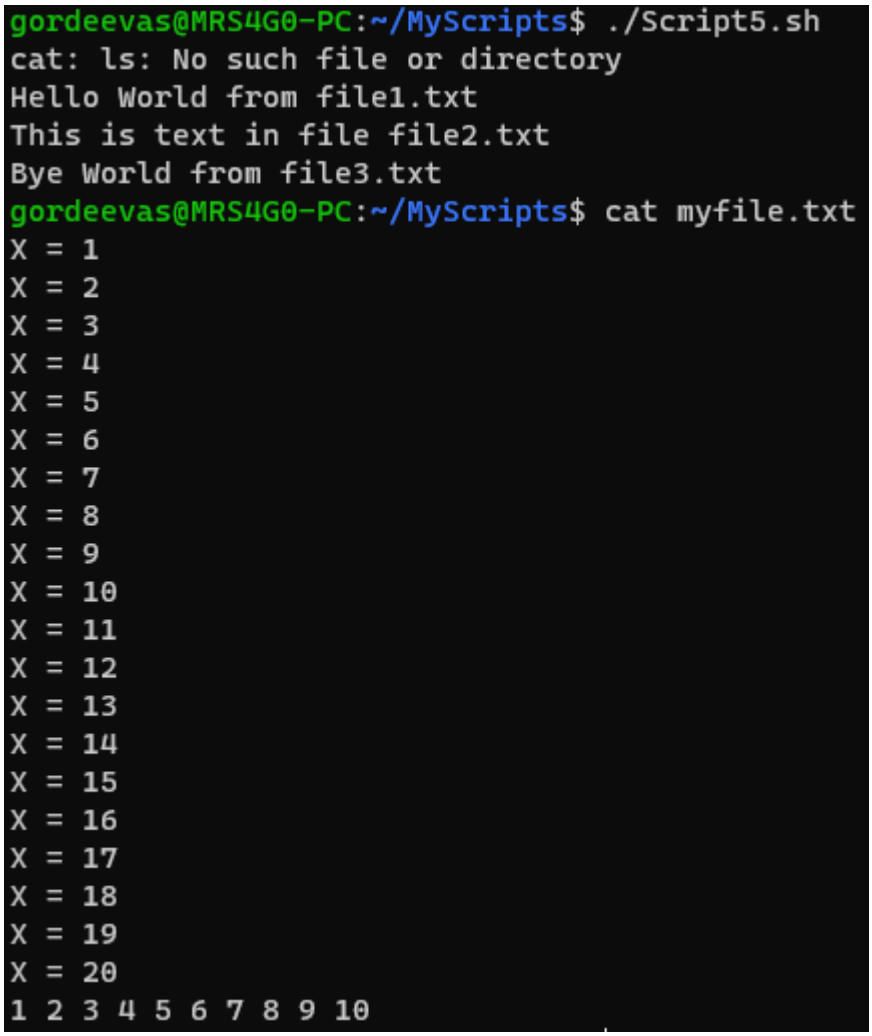
```
gordeevas@MRS4G0-PC:~/MyScripts$ ./Script5.sh
cat: ls: No such file or directory
Hello World from file1.txt
This is text in file file2.txt
Bye World from file3.txt
gordeevas@MRS4G0-PC:~/MyScripts$ ls
MyScript1.sh  MyScript4.sh  Script1.sh  Script4.sh  file2.txt
MyScript2.sh  MyScript5.sh  Script2.sh  Script5.sh  file3.txt
MyScript3.sh  MyScript6.sh  Script3.sh  file1.txt   myfile.txt
gordeevas@MRS4G0-PC:~/MyScripts$ cat myfile.txt
X = 1
X = 2
X = 3
X = 4
X = 5
X = 6
X = 7
X = 8
X = 9
X = 10
```

Модифицируйте bash-скрипт еще раз. Примените его. Изучите результат.

```
#!/bin/bash

for myfile in 'ls *.txt'; do
```

```
    cat $myfile
done

echo -n "" > myfile.txt
for x in {1..20}; do
    echo "X = $x" >> myfile.txt
done

for (( i=1; i<=10; i++ )); do
    echo -n "$i " >> myfile.txt
done
echo "" >> myfile.txt
```

```
./Script5.sh
```



32. Создайте в каталоге “MyScripts” файл “Script6.sh”. В теле bash-скрипта запишите указанный код.

```
nano Script6.sh
```

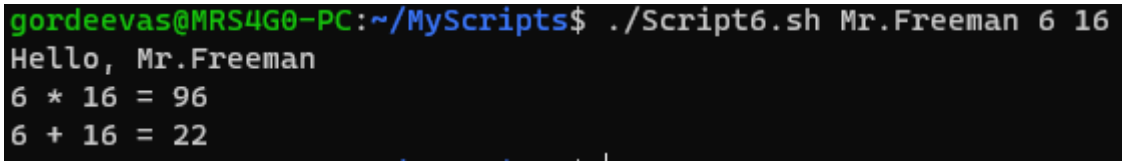
```
#!/bin/bash

sum=0
pr=0
myFunction()
```

```
{
    sum=$(($1+$2))
    pr=$(($1*$2))
}

echo "Hello, $1"

myFunction $2 $3
echo "$2 * $3 = $pr"
echo "$2 + $3 = $sum"
```

```
chmod a+x Script6.sh
```

Сохраните bash-скрипт и примените его, задав 3 входных значения (например, значения “Mr.Freeman”, 6, 16). Изучите результат для того, чтобы понять, как работает код bash-скрипта.

```
./Script6.sh Mr.Freeman 6 16
```



Ответы на контрольные вопросы

1. Зачем в bash-скрипте в первой строке всегда указывают “#!/bin/bash”?

   Это указание на то, что скрипт будет исполняться в bash.

2. К какому результату приводит использование конструкции echo в bashскрипте?

   Выводит строку на экран.

3. Как в bash-скрипте прописать выполнение какой-либо команды, например, ls –l?

   ```
   $(ls -l)
   ```

4. К какому результату приводит выполнение команды chmod a+x?

   Параметр a+x означает, что для всех пользователей будет установлен флаг исполнения.

5. Как в bash-скрипте задать переменную и присвоить ей вывод какой-либо команды?

   ```
   myVar=$(ls -l)
   ```

6. Как в bash-скрипте вывести на экран значение переменной?

   ```
   echo $myVar
   ```

7. Какое значение хранит в себе переменная “$0”?

   Имя скрипта.

8. Как использовать переменные в bash-скрипте так, чтобы им присваивались значения, указанные при выполнении bash-скрипта?

   `./Script1.sh 1 2 3`

9. Как в bash-скрипте выполнять арифметические действия?

   `echo $(($1+$2))`

10. Как в bash-скрипте выполнить команду echo таким образом, чтобы курсор не переходил на строку ниже?

    `echo -n "Hello World"`

11. Как в bash-скриптах работает конструкция if-elif-else-fi? Что такое fi в этой конструкции?

    `if-elif-else-fi` - это конструкция, которая позволяет выполнять различные действия в зависимости от условий. `fi` - это конец конструкции.

12. Какие операции сравнения можно выполнить в конструкции if-elif-else-fi?

    - `-eq` - равно
    - `-ne` - не равно
    - `-gt` - больше
    - `-lt` - меньше
    - `-ge` - больше или равно
    - `-le` - меньше или равно

13. Какой код bash-скрипта позволяет пользователю вводить значения параметров при его (скрипта) выполнении?

    `read myVar`

14. Как работает в bash-скриптах конструкция case-esac? Что такое esac?

    `case-esac` - это конструкция, которая позволяет выполнять различные действия в зависимости от условий. `esac` - это конец конструкции.

15. Как в bash-скриптах реализуется цикл while?

```
while [ $i -le 10 ]; do
    echo $i
    i=$(($i+1))
done
```

16. Как в bash-скриптах реализуется цикл for?

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

17. Как в bash-скриптах можно записать echo в файл?

```
echo "Hello World" >> myfile.txt
```

18. Как задать функцию в bash-скрипте? Как передавать в нее параметры?

```
myFunction()
{
    echo "Hello, $1"
}
```