BASH PROGRAMMING NOTES

1. HELLO BASH SCRIPTING

a) cat /etc/shells :- valid login shells

b) which bash :- path of bash

c) ls -al :- for seeing file permissions

d) chmod +x ./helloScript.sh :- for making a executable file

Ex1:

```
#! /bin/bash
echo “hello bash script”
```

2. REDIRECT TO A FILE

Ex1:

```
#! /bin/bash
echo “hello bash script” > file.txt
```

Ex2:

```
#! /bin/bash
cat > file.txt
```

Ex3:

```
#! /bin/bash
cat >> file.txt
```

3. COMMENTS

Ex1:

```
 # This is a bash command
```

Ex2:

: ‘

This is command 1

This is command 2 ‘

Ex3:

#! /bin/bash

cat << Kreative

This is creative content

Kreative

4. CONDITIONAL STATEMENTS

```
#! /bin/bash
count=10
if [ $count  < 10 ]
then
        echo “$count is less than 10”
elif [[ $count == 10 ]]
then
        echo “$count is equal to 10”
else
        echo “$count is greater than 10”
fi
```

5. LOOPS

a) we can use continue and break statements with if conditions

EX1:

```
#! /bin/bash
number=1
while [ $number < 10 ]
do
        echo “$number”
        number=$(( number + 1 ))
done
```

Ex2:

```
#! /bin/bash
for i in {0..20..2}
do
        echo $i
done
```

6. SCRIPT INPUT

Ex1:

```
#! /bin/bash
echo $1 $2 $3
```

Ex2:

```
#! /bin/bash
args=($@)
echo $@
echo $#
```

Ex3:

```
#! /bin/bash
```

```
while read line
do
        echo “$line”
done < “${1:-/dev/stdin}”
```

7. SCRIPT OUTPUT

Ex1:

```
#! /bin/bash
ls -al 1>file1.txt 2>file2.txt
```

8. SEND OUTPUT FROM ONE SCRIPT TO ANOTHER

Ex1:

```
#! /bin/bash
MESSAGE=”Mandalore is under attack”
export MESSAGE
./secondScript.sh
 And Now the secondScript.sh
#! /bin/bash
echo “The message from the Skywalker given in
helloScript.sh is : $MESSAGE
```

9. STRINGS PROCESSING

a) we can use “\<, \> operators for string manipulation”

b) $str1$str2 for string concatenation

c) ^, ^^ for lower and upper case conversions respectively

```
#! /bin/bash
echo “Enter the first string”
read str1
echo “Enter the second string”
read str2
if [ $str1 == $str2 ]
then
        echo “Strings match”
else
        echo “Strings don’t match”
fi
```

10. NUMBERS AND ARITHMETICS

Ex1:

```
#! /bin/bash
n1=4
n2=5
echo $[ n1 + n2 ]
echo $[n1 - n2 ]
```

Ex2:

```
#! /bin/bash
echo “Enter the Hex number of your choice”
read Hex
echo -n “The decimal value of $Hex is: ”
```

echo “obase=10; ibase=16; $Hex” | bc

We are using here bc calculator and also pipe

11. DECLARE COMMAND

a) declare -p :- gives all variables in system

b) declare temp=20 :- declare temp variable in system

c) # all the commands are given in terminal

12. ARRAYS

a) We can access array elements by using car[index]
b) We can also assign values by car[index]=value

```
#! /bin/bash
car=(‘BMW’  ‘TOYOTA’  ‘AUDI’)
echo “${car[@]}”
```

13. FUNCTIONS

Ex1:

```
#! /bin/bash
function funName()
{
        echo “This is new function”
```

```
}
funName
```

Ex2:

```
#! /bin/bash
function funcCheck
```

14. FILES AND DIRECTORIES

Ex1:

```
#! /bin/bash
mkdir -p Directory
```

Ex2:

```
#! /bin/bash
echo “Enter the directory name to check”
read direct
if [ -d “$direct” ]
then
echo  “$direct Exists”
else
        echo “$direct doesn’t Exists”
fi
```

Ex3:

```
#! /bin/bash
echo “Enter the filename name to check”
read fileName
```

```
if [[ -f “$fileName” ]]
then
        echo   “$fileName Exists”
else
        echo “$fileName doesn’t Exists”
fi
```

Ex4:

```
#! /bin/bash
echo “Enter the filename in which you want to append”
read fileName
if [[ -f “$fileName” ]]
then
        echo   “Enter the text you want to append”
        read fileText
        echo “$fileText” >> $fileName
else
        echo “$fileName doesn’t Exists”
fi
```

Ex5:

```
#! /bin/bash
echo “Enter the filename from which you want to read”
read fileName
if [[ -f “$fileName” ]]
then
        while IFS= read  -r line
```

```
    do
        echo “$line”
    done < $fileName

else
    echo “$fileName doesn’t Exists”
fi
```

a) We can use rm $fileName to remove the file

16. CURL IN SCRIPTS

```
#! /bin/bash

url=”Some url link”
curl ${url} >outputfile
```

17. PROFESSIONAL MENUS

```
select car in BMW TESLA AUDI TOYOTA ROVER
do
    case $car in
    BMW)
        echo "BMW selected";;
    TESLA)
        echo "TESLA selected";;
```

```
        AUDI)
                echo "AUDI selected";;
        TOYOTA)
                echo "TOYOTA selected";;
        ROVER)
                echo "ROVER selected";;
        *)
                echo "Error";;
        esac
done
```

18. INTRODUCTION TO GREP

```
echo "Enter the filename to search text from"
read fileName
if [[ -f $fileName ]]
then
        echo "Enter the text to search"
        read grepvar
        grep -i -n -c $grepvar $fileName
else
        echo "$fileName doesn't exists"
fi
```

19. INTRODUCTION TO AWK

Ex-1:

```
echo "Enter the filename to print form awk"
read fileName
```

```
if [[ -f $fileName ]]
then
        awk '{print}' $fileName
else
        echo "$fileName doesn't exists"
fi
```

Ex-2:

```
echo "Enter the filename to print form awk"
read fileName
if [[ -f $fileName ]]
then
        awk '/some text / {print $2 $3} ' $fileName
else
        echo "$fileName doesn't exists"
fi
```

20. INTRODUCTION TO SED

```
echo "Enter the filename to substitute using sed"
read fileName
if [[ -f $fileName ]]
then
        cat $fileName | sed 's/i/I/g' > newfile.txt
else
        echo "$fileName doesn't exists"
fi
```

21. INTRODUCTION TO BASH DEBUGGING

1. use bash -x ./helloScript.sh
2. use #! /usr/bin/bash -x
3. use set -x to set +x for debugging a code snippet

REFERENCES :

Linuxhint youtube channel :-
https://www.youtube.com/watch?v=e7BufAVwDiM