Day-2

MCA-A

Roll No:- 56

NAME	M. Manjunath	TOTAL MARKS
CLASS	SUBJECT LSS	
ROLL No.	DATE	

→ Programming constructs

Bash shell syntax and constructs

① The shbang line.

The "shbang" line is the very first line of the script and lets the kernel know what shell will be interpreting the lines in the script.

- It consists of #! followed by full path name.

Example:

#!/bin/bash

② Comments

comments are descriptive material preceded by a # sign.

Example:

This is a comment.

③ Local variables

- Local variables are in scope for the current shell. when a script ends they are no longer available. i.e. they go out of the scope.

- Local variables can be defined with built-in declare function.

Eg:- variable_name = value

declare variable_name = value

name = "manju"

x = 5

(4) Global variables

- Global variables are called environment variables and are created with the export built-in command.
- They are set for currently running shell and any process spawned from that shell.
- The built-in declare function with the -x option also sets an environment variable and marks it for export.

Eg:- export VARIABLE_NAME = value
declare -x VARIABLE_NAME = value
export PATH = /bin : /usr/bin :

(5) Extracting values from variables
To extract a value from variables, a dollar sign is used.

Eg: echo $variable_name
echo $name
echo $PATH

NAME		TOTAL MARKS
CLASS	SUBJECT	
ROLL No	DATE	

Q. 2. Looping and Conditional Statements

Conditional Statements

→ The if construct is followed by an expression enclosed in parenthesis.
The operators are similar to C operators.

- The then keyword is placed after the closing parenthesis.
An if must end with an endif.

i) if Statement
This block will process if specified condition is true.

Syntax:
```
if [expression]
then
    Statement
fi
```

Example
```
a=10
b=20
if [ $a == $b ]
then
    echo "a equal to b"
fi
```

ii) if-else Statement
If - specified condition is not true in if part then else part will be executed.

Syntax:
```
if [ expression ]
then
    Statement 1
else
    Statement 2
fi
```

Example:-
```
a= 20
b= 20
if [ $a == $b ]
then
    echo "a equal to b"
else
    echo "a is not equal to b"
```

Switch statement:

- case statement works of a switch statement if specified value match with the pattern then it will execute a block of that particular pattern.

- When a match is found of all of the associates statements until (;;) double semicolon is executed. If there is no match, the exit status of case is zero.

Syntax

```
case in
    pattern 1) statement 1;;
    pattern n) statement n;;
esac
```

Example:

```
CARS="bmw"

# pass the variable in string
case "$cars" in
    # case 1
    "mercedes") echo "Headquater - Germany" ;;

    # case 2
    "audi") echo "Headquaters - India" ;;

    # case 3
    "bmw") echo "Headquater - Italy" ;;
```

output:

```
$ bash -f file1.sh
Headquater - Italy
```

NAME		TOTAL MARKS
CLASS	SUBJECT	
ROLL No.	DATE	

→ Looping statements

There are 3 looping statements

1. While statement

Here command is evaluated and boled on the result loop will executed.

If command raise to false then loop will be terminated.

Syntax:

```
while [expression]
do
    statements
done
```

Example

```
a=0
# -lt is lese than operator
while [ $a -lt 10 ]
do
    echo $a
# increment the value
    a=`expr $a + 1`
```

2. for statements

The for loop operates on lists of items.

It repeats a set of commands for every item in list.

Syntax:

```
for item_list
do
    statements
done
```

Example:

```
for a in 1 2 3 4 5 6 7
do
    if [ $a == 5 ]
    then
        break
    fi
    echo "Iteration no $a"
done
```

- until statement

The until statement is executed as many as times the condition/command evaluates to false.
The loop terminates when the condition becomes true

Syntax

```
until [expression]
do
    statements
done
```

Example:

```
a=0
until [ $a -gt 10 ]
do
    echo $a
    a=`expr $a +1`
done
```