SHELL SCRIPTING

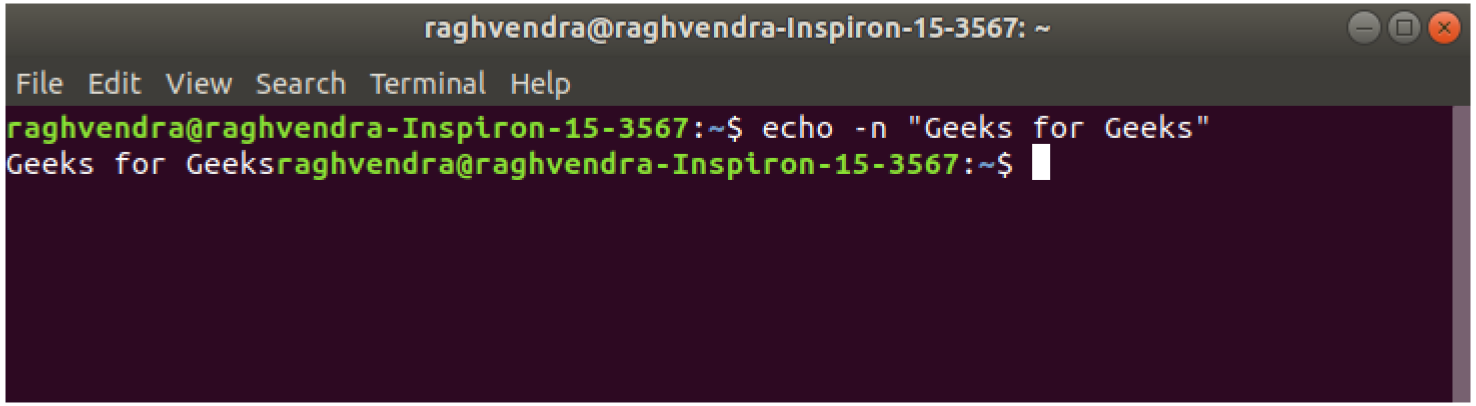
Shell types

- The Bourne Shell
- The C Shell
- The Korn Shell
- The GNU Bourne-Again Shell

- The expr command is used to perform arithmetic operations.
- Syntax:
- expr op1 math-operator op2

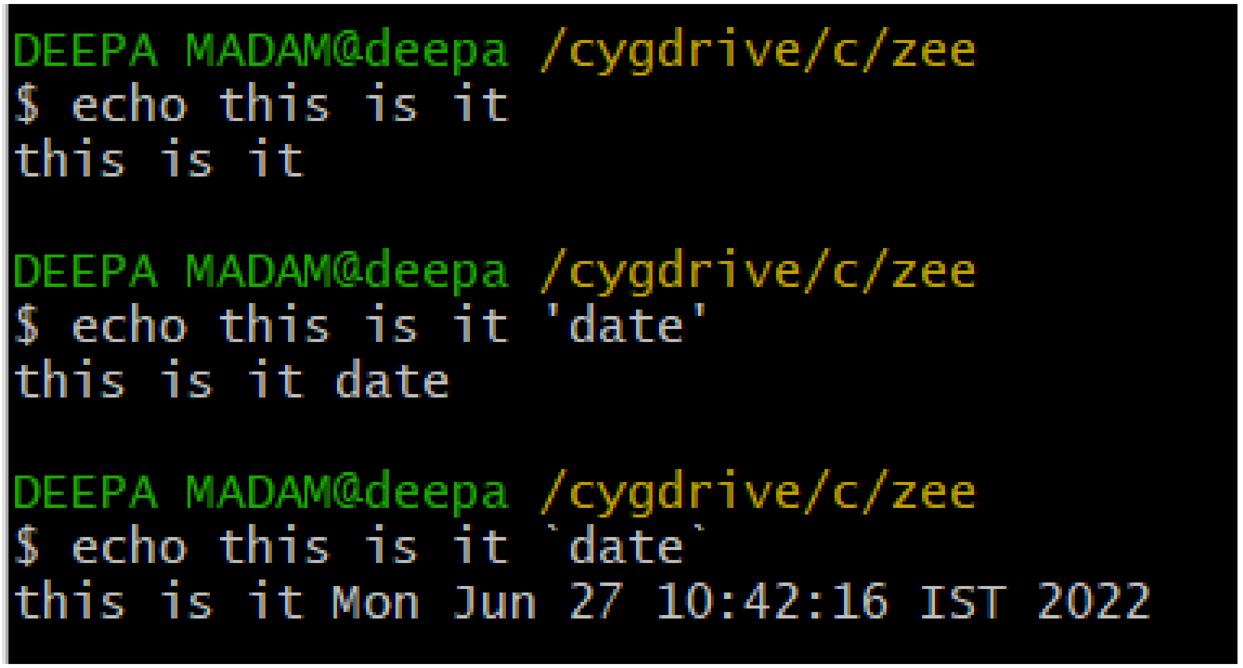
- $ expr 1 + 3
- $ expr 2 - 1
- $ expr 10 / 2
- $ expr 20 % 3
- $ expr 10 * 3
- $ echo ‘expr 6 + 3‘

- **echo** command in linux is used to display line of text/string that are passed as an argument .
- **-e** here enables the interpretation of backslash escapes
- **-n :** this option is used to omit echoing trailing newline

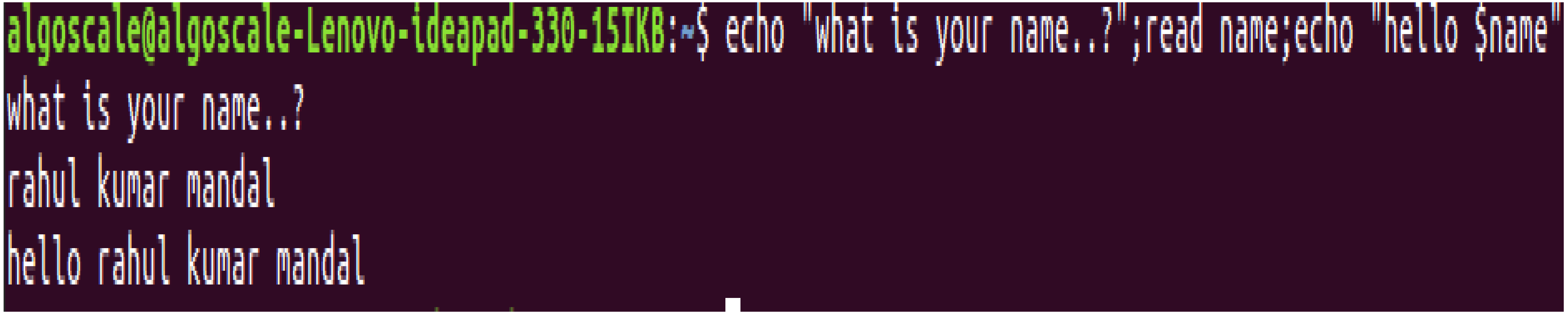

raghvendra@raghvendra-Inspiron-15-3567: ~
File Edit View Search Terminal Help

```
raghvendra@raghvendra-Inspiron-15-3567:~$ echo -n "Geeks for Geeks"
Geeks for Geeksraghvendra@raghvendra-Inspiron-15-3567:~$ 
```

Quotes	Name	Meaning
"	Double Quotes	"Double Quotes" - Anything enclose in double quotes removed meaning of that characters (except \ and $).
'	Single quotes	'Single quotes' - Enclosed in single quotes remains unchanged.
`	Back quote	`Back quote` - To execute command

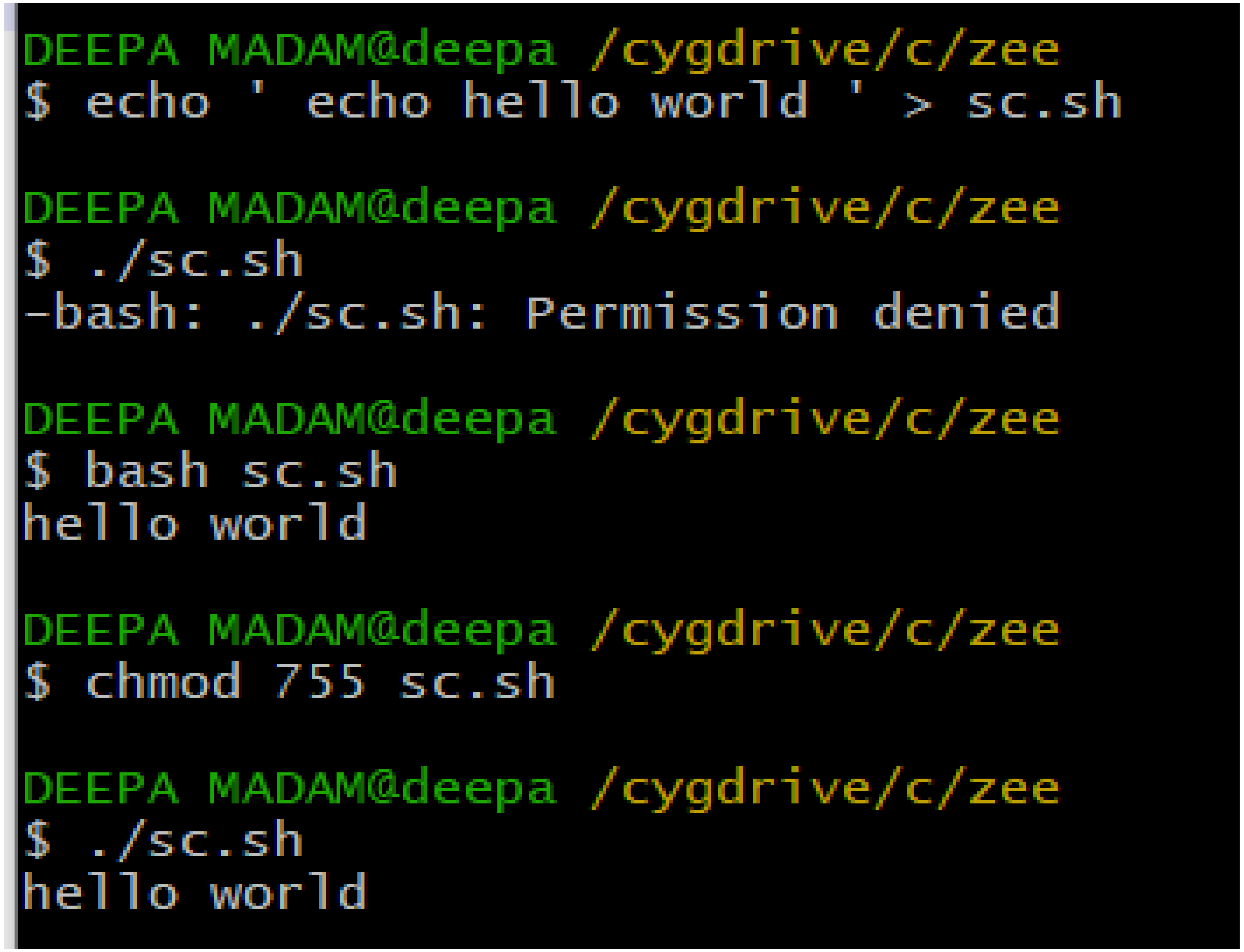


```
DEEPA MADAM@deepa /cygdrive/c/zee
$ echo this is it
this is it

DEEPA MADAM@deepa /cygdrive/c/zee
$ echo this is it 'date'
this is it date

DEEPA MADAM@deepa /cygdrive/c/zee
$ echo this is it `date`
this is it Mon Jun 27 10:42:16 IST 2022
```



```
algoscale@algoscale-Lenovo-ideapad-330-15IKB:~$ echo "what is your name..?";read name;echo "hello $name"
what is your name..?
rahul kumar mandal
hello rahul kumar mandal
```

Wild card /Shorthand	Meaning	Examples	
*	Matches any string or group of characters.	$ ls *	Will show all files
		$ ls a*	Will show all files whose first name is starting with letter 'a'
		ls *.c	Will show all files having extension.c
		$ ls ut*.c	Will show all files having extens.c but file name must begin with 'ut'.
?	Matches any single character.	$ ls ?	Will show all files whose names are1 character long
		$ ls fo? with fo	Will show all files whose names are 3 character long and file name begin
[...]	Matches any one of the enclosed characters	$ ls [abc]*	Will show all files beginning with letters a,b,c



```
DEEPA MADAM@deepa /cygdrive/c/zee
$ echo ' echo hello world ' > sc.sh

DEEPA MADAM@deepa /cygdrive/c/zee
$ ./sc.sh
-bash: ./sc.sh: Permission denied

DEEPA MADAM@deepa /cygdrive/c/zee
$ bash sc.sh
hello world

DEEPA MADAM@deepa /cygdrive/c/zee
$ chmod 755 sc.sh

DEEPA MADAM@deepa /cygdrive/c/zee
$ ./sc.sh
hello world
```

System Variable	Meaning	To View Variable Value Type
BASH=/bin/bash	Our shell name	echo $BASH
BASH_VERSION	Holds the version of this instance of bash.	echo $BASH_VERSION
COLUMNS=80	No. of columns for our screen	echo $COLUMNS
HOSTNAME	The name of the your computer.	echo $HOSTNAME
CDPATH	The search path for the cd command.	echo $CDPATH
HISTFILE	The name of the file in which command history is saved.	echo $HISTFILE
HISTFILESIZE	The maximum number of lines contained in the history file.	echo $HISTFILESIZE
HISTSIZE	The number of commands to remember in the command history. The default value is 500.	echo $HISTSIZE
HOME	The home directory of the current user.	echo $HOME
IFS	The Internal Field Separator that is used for word splitting after expansion and to split lines into words with the read builtin command. The default value is <space><tab><newline>.	echo $IFS
LANG	Used to determine the locale category for any category not specifically selected with a variable starting with LC_.	echo $LANG

LINES=25	No. of columns for our screen	echo $LINES
PATH	The search path for commands. It is a colon-separated list of directories in which the shell looks for commands.	echo $PATH
PS1	Your prompt settings.	echo $PS1
TMOUT	The default timeout for the read builtin command. Also in an interactive shell, the value is interpreted as the number of seconds to wait for input after issuing the command. If not input provided it will logout user.	echo $TMOUT
TERM TERM=vt100	Your login terminal type.	echo $TERM export
SHELL	Set path to login shell.	echo $SHELL
DISPLAY	Set X display name	echo $DISPLAY export DISPLAY=:0.1
EDITOR	Set name of default text editor.	export EDITOR=/usr/bin/ vim
OSTYPE=Linux	Our Os type	echo $ OSTYPE
LOGNAME=students	students Our logging name	echo $ LOGNAME
USERNAME=rshukla	User name who is currently login to this PC	echo $ USERNAME

- variable names –
- _ALI
- TOKEN_A
- VAR_1
- VAR_2

- VAR1=“abc"
- VAR2=100

Readonly variable

- NAME="Zara Ali"
- readonly NAME
- NAME="Qadiri"

- The above script will generate the following result −
- /bin/sh: NAME: This variable is read only.

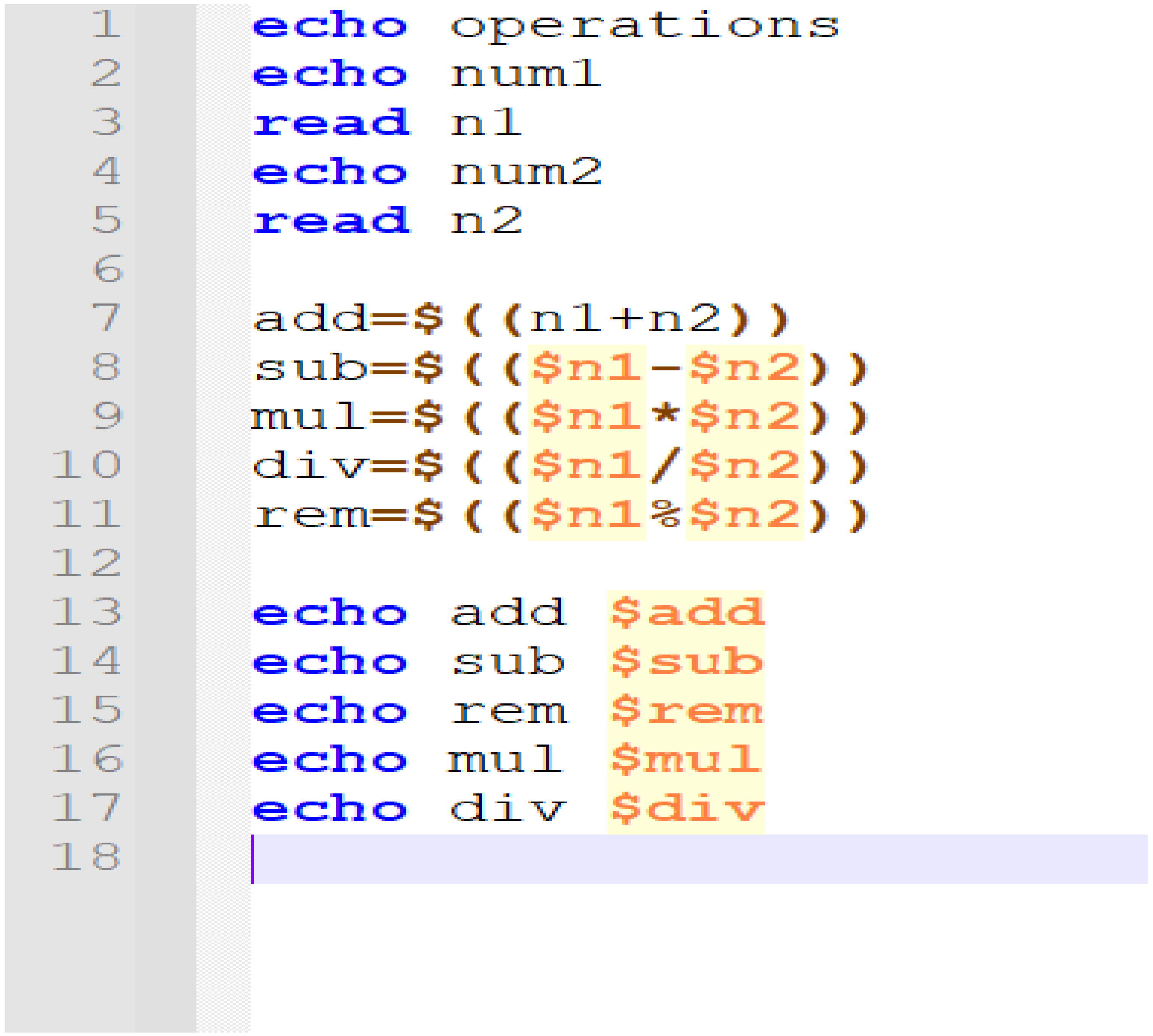
unset variable

- NAME="Zara Ali"
- unset NAME
- echo $NAME

SPECIAL SHELL VARIABLES

- echo "File Name: $0“
- echo "First Parameter : $1“
- echo "Second Parameter : $2“
- echo "Quoted Values: $@“
- echo "Quoted Values: $*“
- echo "Total Number of Parameters : $#"
- echo “The exit status of the last command executed : $?”
- echo “process number of the current shell : $$”

Arithmetic operator



```
echo operations
echo num1
read n1
echo num2
read n2

add=$((n1+n2))
sub=$(($n1-$n2))
mul=$(($n1*$n2))
div=$(($n1/$n2))
rem=$(($n1%$n2))

echo add $add
echo sub $sub
echo rem $rem
echo mul $mul
echo div $div

```

Operator	Description	Example
-eq	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.	[$a -eq $b] is not true.
-ne	Checks if the value of two operands are equal or not; if values are not equal, then the condition becomes true.	[$a -ne $b] is true.
-gt	Checks if the value of left operand is greater than the value of right operand; if yes, then the condition becomes true.	[$a -gt $b] is not true.
-lt	Checks if the value of left operand is less than the value of right operand; if yes, then the condition becomes true.	[$a -lt $b] is true.
-ge	Checks if the value of left operand is greater than or equal to the value of right operand; if yes, then the condition becomes true.	[$a -ge $b] is not true.
-le	Checks if the value of left operand is less than or equal to the value of right operand; if yes, then the condition becomes true.	[$a -le $b] is true.

Operator	Description	Example
!	This is logical negation. This inverts a true condition into false and vice versa.	[! false] is true.
-o	This is logical OR. If one of the operands is true, then the condition becomes true.	[$a -lt 20 -o $b -gt 100] is true.
-a	This is logical AND. If both the operands are true, then the condition becomes true otherwise false.	[$a -lt 20 -a $b -gt 100] is false.

```
if [ expression ]
then
Statement(s) to be executed if expression is true
fi
```

```
if [ expression ]
then
Statement(s) to be executed if expression is true
else
Statement(s) to be executed if expression is not true
fi
```

```
if [ $n1 -gt $n2 ]
then
echo "$n1 is greater“
else
echo "$n2 is greater“
fi
echo "done"
```

```
if [ expression 1 ]
Then
Statement(s) to be executed if expression 1 is true
elif [ expression 2 ]
Then
Statement(s) to be executed if expression 2 is true
elif [ expression 3 ]
Then
Statement(s) to be executed if expression 3 is true
Else
Statement(s) to be executed if no expression is true
fi
```

```
echo enter weekday
read w
if [ $w -eq 1 ]
then
echo "monday“
elif [ $w -eq 2 ]
then
echo "tuesday“
elif [ $w -eq 3 ]
then
echo "wed“
else
echo "wrong“
fi
```

```
case word in
pattern1)
Statement(s) to be executed if pattern matches
;;
pattern2)
Statement(s) to be executed if pattern2 matches
;;
pattern3)
Statement(s) to be executed if pattern3 matches
;;
*)Default condition to be executed
;;
esac
```

```
case $num in
1 )
echo "one“
;;
2 )
echo "two“
;;
3 )
echo "three“
;;
* )
echo wrong
;;
esac
```

While loop

```
while command
do
Statement(s) to be executed if command is true
done
```

```
a=0
while [ $a -lt 10 ]
do
echo $a
a=`expr $a + 1`
done
```

```
while [ $n -lt 11 ]
do
echo "$n * $v = $(($n*$v))"
n=$((n+1))
done
```

```
until command
do
Statement(s) to be executed until command is true
done
```

```
n=4
until [ $n -gt 11 ]
do
echo $n
n=$((n+1))
done
```

```
for var in 0 1 2 3 4
do
echo $var
done
echo for done
```

```
for var in {1..20..3}
do
echo
$var
done
echo for done
echo enter number
read a
for((i=10; i>=$a; i--))
do
echo $i
done
```

- select var in word1 word2 ... wordN
- do
- Statement(s) to be executed for every word.
- done

```
select DRINK in tea cofee water juice appe all none
do
case $DRINK in
tea|cofee|water|all)
echo "Go to canteen"
;;
juice|appe)
echo "Available at home"
;;
none)
break
;;
*)
echo "ERROR: Invalid selection"
;;
esac
done
```

QUESTIONS

- Write a shell program to add two numbers.
- Write a shell program to take user input.
- Write a shell program to perform addition, subtraction, multiplication and division.
- Write a shell program to check whether the number is even or odd.
- Write a shell program to swap two numbers.
- Write a shell program to find the largest in two numbers.
- Write a shell program to find the largest in three numbers.
- Write a shell program for the temperature converter.
- Write a shell program for a currency converter.
- Write a shell program to check whether the number is palindrome or not.
- Write a shell program to print even sequence.
- Write a shell program to print odd sequence.
- Write a shell program to print the Fibonacci series.
- Write a shell program to print the factorial of a number.
- Write a shell program to check whether the number is prime or not.