GETTING STARTED
WITH
TEST COMMANDS

➢ THE TEST COMMANDS:

- The test command investigates the sort of tests that we raised above and then it translates the result into the language of success or failure.
- This helps the shell in deciding whether to execute the commands in the if block or the commands in the else block.
- The test command can carry out several types of tests these are:
  - File Tests
  - String Tests

FILE TESTS:

- The test command has several options for checking the status of a file. Using file tests, we can find out whether the specified file is an ordinary file or a directory, or whether is grants read, write or execute permissions, so on and so forth.
- The file test commands are given as follow:

-s : True, If file is not an Empty (Size is greater than 0).
-f : True, If it is a Regular file.
-d : True, If it is a Directory.
-l : True, If it is a Link file.
-c : True, If it is a Character special file.
-b : True, If it is a Block special file.
-k : True, If it is a Sticky bit is set.
-r : True, If it is a Read permission.
-w : True, If it is a Write permission.
-x : True, If it is a Execute Permission.

- Now how we can use a file test in a shell script:

1. **Testing a File:**

```
#!/bin/sh
echo "Enter a file name:"
read fname
if [ -f $fname ]
then
echo "You indeed entered a file name"
else
echo "What you entered $fname is not a file."
fi
```

2. **Testing a Directory:**

```
#!/bin/sh
echo "Enter a file name:"
read fname
if [ -d $fname ]
then
echo "You indeed entered a directory name"
else
echo "What you entered $fname is not a Directory"
fi
```

3. **This one check whether the user has a write permission to a file:**

```
#!/bin/sh
echo "Enter a file name:"
read fname
if [ -w $fname ]
then
echo "Type matter to append. To stop type ctrl+c"
cat>>$fname
else
echo "No write permission"
fi
```

STRING TESTS:

- Another set of tests that the test command can handle are the string tests.

String1 = String2	**:** True, If strings are same.
String1 != String2	**:** True, If strings are different.
-z	**:** True, If the length of the string is Zero.
-n	**:** True, If string is not empty (Greater than 0)
string	**:** True, If the string is not a null string.

- Now how we can use a string test in a shell script:

1. String Test Examples:

```
#!/bin/sh
string1=Good
string2=Bad
string3=
[ $string1 = $string2 ]
echo $?
[ $string1 != $string2 ]
echo $?
[ -n $string1 ]
echo $?
[ -z "$string3" ]
echo $?
[ -z $string3 ]
echo $?
["$string3 ]
echo $?
```

How do you know whether the two strings being compared are identical or not, or whether length of a string is zero or non-zero? that's the time the meta character $? comes to help. This meta character contains the success or failure of the last command that has been executed. That's the reason we have used a separate echo statement after each test. each echo statement reports the Success (0) or failure (1) of the test preceding it.

NESTED IF-ELSES:

- It is perfectly all right if we write an entire if-else-fi construct within either the body of the if statement or the body of an else statement.
- This is called 'nesting' of **ifs.**

```
#!/bin/sh
echo "Enter either 1 or 2"
read i
if [ $i -eq 1 ]
then
echo "You would go to Heaven!"
else
```

```
if [ $i -eq 2 ]
then
echo "hell was created with you in mind"
else
echo "How about mother earth!"
fi
fi
```

1. **Program accept a file name & check the file is Regular or directory file:**

```
#!/bin/sh
echo -n "Enter a file name:"
read fname
if [ -e $fname ]
then
if [ -f $fname ]
echo "$fname is a Regular file"
elif [ -d $fname ]
echo "$fname is Directory file"
then
echo "It is not a File or Directory"
fi
else
echo "$fname file doesn't exist"
fi
```

2. **Script accept 2 filenames and check the given 2 files are same / not:**

```
#!/bin/sh
echo -n "Enter a file name 1:"
read fname1
echo -n "enter a filename 2:"
read fname2
x=`cmp $fname1 $fname2`

if [ -z "$x" ]
then
echo "Given 2 files are same"
else
echo "Given 2 files are not same"
fi
```

3. **Write a script accept a string and check the given string is empty or not:**

```
#!/bin/sh
echo -n "enter a string:"
read string
if [-z "$string" ]
then
echo "Given string empty"
else
echo "Given string not empty"
fi
```

4. **Write a script accept a single character and check the given character is Alphabet or Digit or Special character:**

```
#!/bin/sh
echo -n "Enter a single character:"
read char
case $char in
[a-zA-Z])echo "Alphabet";;
[0-9])echo "Digit";;
[^a-zA-Z0-9])echo "Special Character";;
*)echo "You entered more than one character";;
Esac
```

5. **Write a script accept a single character and check the given character is special character or digit or vowel or consonant:**

```
#!/bin/sh
echo -n "Enter a single character:"
read char
case $char in
[^a-zA-Z0-9])echo "Special character";;
[0-9])echo "Digit";;
[AEIOUaeiou])echo "Vowel";;
[^AEIOUaeiou])echo "Consonant"
*)
echo "You entered more than one character";;
esac
```