UNIT - 4

(1)

Page No.:
Date: / /

Shell syntax :-

Variables :-

→ It is not necessary to declare variable before using them.

→ ~~screen~~ we create them by assigning values to them.

→ By default all variables are considered and stored as string. even when they are assign string values.

→ Linux is case sensitive.

→ Contents of a variable can be extracted by using $ sign.

for eg. $sname.

Quoting :- (use of quotes)

→ Whenever we want a parameter to have spaces we must quote the parameter.

→ double quotes doesn't affect the substitution of the variable while single quotes and backslash do.

→ we use 'read' command to get string from user.

CONDITIONS :-

A shell script can test the exit command that can be invoked from the command line. That's why it is important to include an exit command with a value at the end of any script.

The test or [command

→ The test or [is same, except when [is used a trailing] is also used for readability.

```
if test -f prgl.txt
then
- - - - - -
- - - - - -
fi
```

```
if [ -f prgl.txt ]
then
=
=
=
fi
```

separator

```
if [ -f prgl.txt ] ; then
=
=
=
fi
```

→ The conditions that we use with test command falls into three different categories -

a) string comparision.

a) string1 = string2 true if equal

b) string1 != string2 true if Not equal

c) -n string true if Not null

d) -z string true if null

(B) Arithmetic comparison

expr1	-eq	expr2	true if equal
expr1	-ne	expr2	true if not equal
expr1	-gt	exp2	true if expr1 > expr2
expr1	-ge	expr2	true if expr1 ≥ exp2
expr1	-lt	expr2	true if expr1 < exp2
expr1	-le	expr2	true if expr1 ≤ expr2
! expr			if expression is false

[C] file condition

-d file true if file is a directory

-e file true if file exist

-f file true if file is a regular file

-g file true if set-group-id is set on file

Page No. :
Date : / /

for

syntax :-

```
for variable in values
do
    statements
done
```

```
for file in $(ls f*.sh)
do
   lpr $file
done
exit 0
```

It will print all the script files starting with f in the current directory and all script end in sh.

while

```
while condition do
    statements
done
```

```
echo " enter password"
read trythis
while [ "$trythis" != "secret" ]
do
echo " sorry, try again"
read trythis
done
    exit 0
```

execution :-

```
Enter password
xyz.
sorry, try again
secret
$
```

untill

syntax :-

```
untill condition
do
    statement
done
```

```
until who | grep "$1" > /dw/null
do
    sleep 60
done
```

File checking :-

General format is -

test operator <file-name>

where the operator can be

-e	true if file exist
-r	true if file is a regular file
-d	true if directory
-r	true if file is readable
-w	true if file is writeable
-x	true if file is executable
-s	true if size is greater than zero size

Example :-

```
if test -r 9.c
then
    echo the file -9.c exists
        and it is readable
fi
```

we can combine expressions using following logical operator

! NOT
It is a unary operator that negates the result of specified expression.

-a AND operation
Returns true if both expressions are true.

-o OR operation

Returns false if both are false. If any one is true the result is true.

case statement

General format is -

```
case value in

pattern1)   < commands > ;;
pattern2)   < commands > ;;
    :
patternN)   < commands > ;;
esac
```

command execution.

→ Positional parameters are given below:-

$0 Refers to the name of the command

$1 Refers to the first argument

$2 Refers to the second argument

$* Refers all the arguments

$# Refers the total no. of arguments

$? Returns the exit status of the last executed command

If a command is executed successfully without giving any error message then exit status of that command is zero.

$! Returns PID of the last background command

$$ Returns PID of the current the current shell

Command Line arguments

command line arguments (also klas positional parameters) are the arguments specified at the command prompt with a command or script to be executed.

The location at the command prompt of the arguments as well as the location of the command, or the script itself are stored to corresponding variables.

These variables are specified shell variable.

UNIT - 4th.

Arithmetic in shell script

shell script variables are by default treated as string not numbers which add some complexity to doing math in shell script.

To keep with script programming paradigm and allow for better math support, languages such as perl or python could be better suited when math is desired.

However it is possible to do maths with shell script. In fact over the years multiple facilities have been added to unix to support working with numbers.

Declare

By using declare statement we can declare a variable.

for example —

```
$ n=6/3
$ echo $n
```

O/P 6/3

$ declare -i n

$ n=6/3

$ echo $n

o/p 2

→ When we don't declare a variable it is treated as string but if we declare a variable then it is not treated as string.

- For example :-

In the above example, when n is not declare, by default it is treated as string so o/p is

6/3

→ But we declare n as integer i.e

-i n

then the o/p is | 2 |

Light Green
Date
Page

EXIT STATUS OF A COMMAND.

→ Every linux command executed by the shell script or user has an exit status.

→ The exit status is an integer no. It takes value from (0 - 255).

→ The linux man pages state the exit status of each command.

→ exit status '0' means the command was executed successfully.

→ A non-zero exit status means command was not executed successfully.

values (1 - 255) shows reasons for command failure.

→ We can use special shell variable ? to get the exit status of previously executed command.

→ echo $?
→ Date
→ echo $?
- If required we can store this value

Light Green
Date
Page

status = $?

sleep and wait

(1) SLEEP

This command is used in shell script to delay the execution.

general format is —

sleep <Number of seconds>

This command will delay execution for specified number for seconds.

for example :-

```
$ echo hello
  sleep 60
  echo "Hi"
```

The string "Hi" will be displayed 1 min later of the display hello.

(2) WAIT

→ This command is used to suspend execution till child process completed.

→ wait is a built-in command of linux that waits for completing running process.

→ wait command is used with a particular process-id or job-id.

- when multiple processes are running in the shell then only pid of last command will be known by the current shell. If wait command is executed this time, then it will be applied for the last command.

- If no pid is given then it will wait for all current child process to complete and returns exit status.

general format is -

wait <pid>

Example :-

wait 120

where 120 is pid.

Light Green
Date
Page

SCRIPT TERMINATION

→ We can use exit statement to indicate successful or unsuccessful shell script termination.

→ The value of N can be used by other commands or shell scripts to take their own action.

→ If we omitte N, the exit status shows the exit status of last command executed.

→ The syntax is as follows :-

```
exit N
```

→ The exit statement is used to exit from the shell script with a status of N.

→ Use the exit statement to terminate shell script upon an error.

→ for example :-

```
echo "Hi I am fine"
# terminate shell script
exit 0
```

* * *

SHELL METACHARACTERS

In linux we can use special characters that the shell interprets rather than passing to the command.

These are called shell metacharacters.

The shell metacharacters are listed below -

SYMBOL	MEANING
>	o/p redirection
>>	o/p redirection append
<	i/p redirection
*	File substitution wildcard. zero one more character
?	File substitution character only one character
[]	any character bet^n brackets.
`cmd`	command substitution
$(cmd)	command substitution
|	pipe

Light Green
Date
Page

; sequence of commands.

|| OR

&& AND

& run command in background

comment-

$ Expand the value of a variable

\ prevent- interpretation of next- character.

How to avoid shell Interpretation of metacharacter.

we can do this by -

1) Escape the metacharacter by backslash (\).

for example :-

echo "\$9.800"

2) use single quote around a string.

Light Green
Date
Page

3) use double quotes. Double quote protects all characters except the backslash.

SHELL MISCELLANY