



Bash Scripting

Ferdowsi University of Mashhad Linux Users Group
t.me/FUMLUG

\$ Definition

Scripts are lines of commands executing in order of appearance

- Scripts can automate jobs and save your time
- Scripts save your fingers !

Example:

```
#!/bin/bash
echo Your username is "$USER"
echo Now I print a list of current directory: $PWD
echo "*****"
ls
```

\$ Step Through

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

Shebang #!

- Determines interpreter of a script
- Appears only once at the first line
- Guarantees consistency since different shells have different syntax

Find Bash Interpreter:

```
$ which bash
```

\$ Variables

Any programming language needs variables

- You define a variable as follows:

```
X="Hello"
```

- You access a variable as follows:

```
$X
```

- Don't put whitespace between elements
- You need quotations if value contains whitespace



```
X = Hello
```



```
X=Hello
```



```
X=Hello World
```



```
X="Hello World"
```

\$ Single vs Double Quotes

Double Quotes (" "):

- Less predictable
- Less flexible
- Needs escape for certain characters

Single Quotes (' '):

- Completely Predictable
- Less flexible
- Needs escape for certain characters

Difference:

```
X="$USER"  
echo $X          # shadow_m2
```

```
X='$USER'  
echo $X          # $USER
```

\$ Input And Comments

You can receive data from user as follows:

```
read X  
echo $X
```

You can have comments as follows:

```
# This is a comment  
# Only single line comments are allowed
```

Question: Input a directory then print a sorted list of it's content

```
#!/bin/bash  
read dir  
ls "$dir" | sort  
# variable should be enclosed in double quotation
```

\$ Conditional Statements

Sometimes, it's necessary to check for certain conditions. Does a string have 0 length ? does the file "foo" exist, and is it a symbolic link , or a real file ? Firstly, we use the if command to run a test. The syntax is as follows:

Test Command

- The command used in conditionals nearly all the time is the test command. Test returns true or false (more accurately, exits with 0 or non zero status) depending respectively on whether the test is passed or failed. It works like this:

```
test operator1 operand operator2
```

- Also used like this:

```
[ operator1 operand operator2 ]
```

```
if condition
then
    statement1
    statement2
    .
    .
else
    statement1
    statement2
    .
    .
fi
```

\$ Conditional Statements Contd.

Arithmetic Operators

-lt	<
-gt	>
-le	<=
-ge	>=
-eq	==
-ne	!=

Unary Operators

-n	operand non zero length
-z	operand has zero length
-d	there exists a directory whose name is operand
-f	there exists a file whose name is operand

\$ Arithmetic Operation

Enclose expressions in (()) to calculate them, like this:

```
a=$(( 22 + 33 ))  
b=$((a*10))  
c=$((b/a))      # No floating point
```

Arithmetic comparison can be done this way and used in conditional statements

```
#!/bin/bash  
read age  
if ((age > 50))  
then  
    echo "Too old"  
else  
    echo "Too young"  
fi
```

\$ While Loop

While loops iterate "while" a given condition is true

Example:

```
#!/bin/bash
X=0
while [ $X -le 20 ]
do
    echo $X
    X=$((X+1))
done
```

C Language Equivalent:

```
for (int X=0,X<=20; X++)
    printf("%d\n", X);
```

\$ For Loop

For loops iterate on a list of data. syntax is as follows:

```
for variable in list
do
    statement1
    statement2
    .
    .
    .
done
```

* Values (including lists) can be substituted by commands

```
files=`find . -type f`
```

\$ Pass Arguments

If your script receives arguments, you can refer to them according to their position: \$1, \$2, \$3...

Example: safe copy

```
#!/bin/bash
if [ -f "$1" ]; then
    if [ -d "$2" ] then
        cp "$1" "$2"
    else
        mkdir "$2"
        cp "$1" "$2"
    fi
fi
```

Example: Single line calculator

```
#!/bin/bash
echo "$(($1 $2 $3))"
```

\$ Question

* Determine changes to files and directories of input directory !

Hints:

- Access to input

```
"$1"
```

- Define Variable

```
varname="value"
```

- Equality Operator

```
[ "$var1" -eq "$var2" ]
```

- Number of files and directories

```
find "$1" | wc -l
```

```
Store number in var1
Define another var2=var1
while loop ( var1 = var2 )
do
    sleep 5
    var2=new number
done
echo "Something's fishy !"
```