PW SKILLS

Bash control structures (if-else loops)

Lesson Plan

Bash control structures(if-else loops)

Bash control structures, such as if-else and loops, are essential for controlling the flow of a script. They allow you to execute different blocks of code based on certain conditions and repeat tasks as needed. Here's a detailed explanation of these control structures in Bash:

1. If-Else Statements

The if-else statement is used to execute a block of code if a specific condition is true. If the condition is false, you can execute an alternative block of code using else.

Let us have a look at the syntax of the if-else condition block.

```
if [condition]
then
   statement1
else
   statement2
fi
```

Here we have four keywords, namely if, then, else and fi.

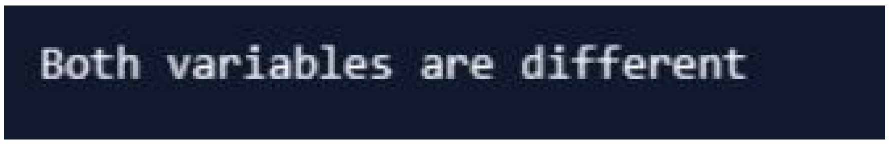
- The keyword **if** is followed by a condition.
- This condition is evaluated to decide which statement will be executed by the processor.
- If the condition evaluates to **TRUE**, the processor will execute the statement(s) followed by the keyword then. In the syntax, it is mentioned as statement1.
- In a case where the condition evaluates to **FALSE**, the processor will execute the statement(s) followed by the keyword else. This is denoted as statement2 in the function syntax.

Here are some useful examples of if-else in shell scripts to give you a better idea of how to use this tool.

1. Using if-else to check whether two numbers are equal:

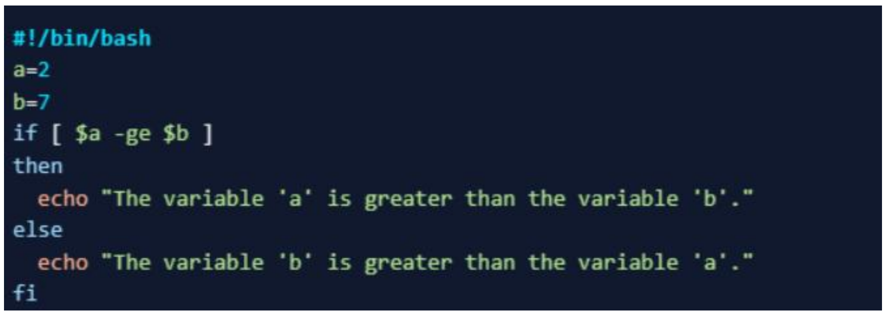
```
#!/bin/bash
m=1
n=2

if [ $n -eq $m ]
then
        echo "Both variables are the same"
else
        echo "Both variables are different"
fi
```

Output:



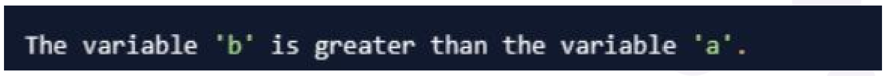
```
Both variables are different
```

2. Using if-else to compare two values:



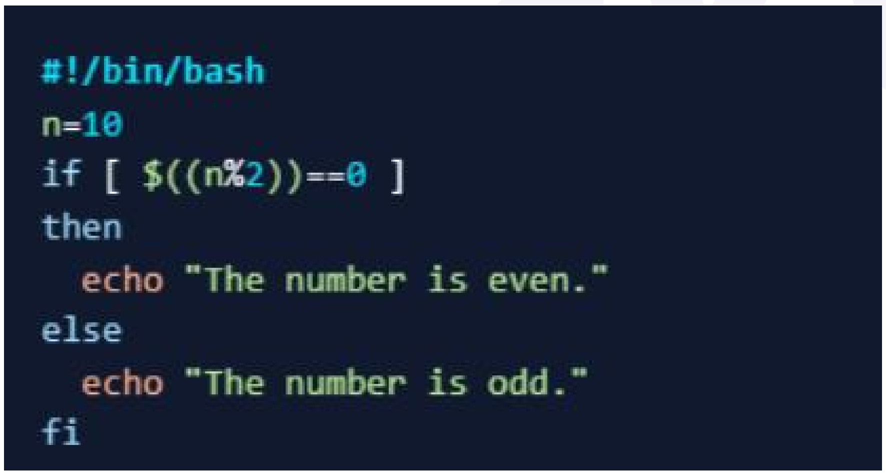
```
#!/bin/bash
a=2
b=7
if [ $a -ge $b ]
then
  echo "The variable 'a' is greater than the variable 'b'."
else
  echo "The variable 'b' is greater than the variable 'a'."
fi
```

Output:



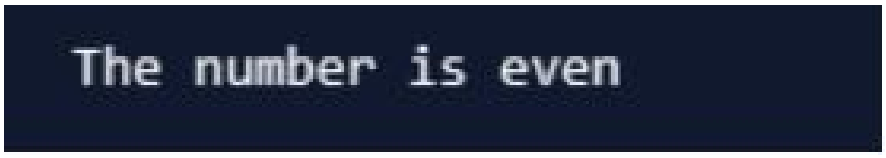
```
The variable 'b' is greater than the variable 'a'.
```

3. Using if-else to check whether a number is even:



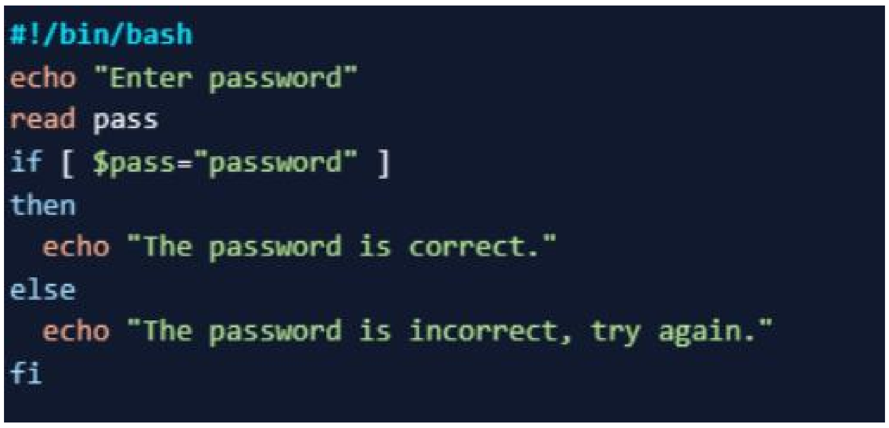
```
#!/bin/bash
n=10
if [ $((n%2))==0 ]
then
  echo "The number is even."
else
  echo "The number is odd."
fi
```

Output:

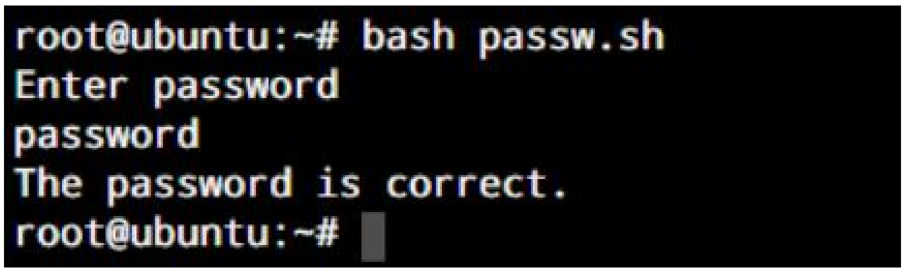


```
The number is even
```

4. Using if-else as a simple password prompt:



```
#!/bin/bash
echo "Enter password"
read pass
if [ $pass="password" ]
then
  echo "The password is correct."
else
  echo "The password is incorrect, try again."
fi
```



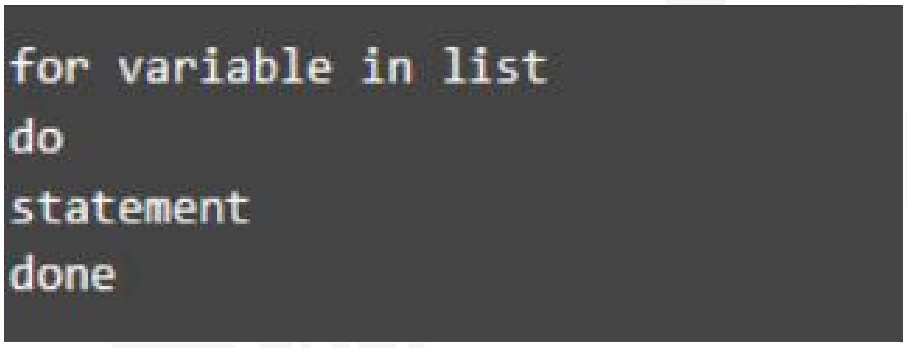
```
root@ubuntu:~# bash passw.sh
Enter password
password
The password is correct.
root@ubuntu:~#
```

2. Loops
Loops in Bash are used to repeat a block of code multiple times.

Types of Loops
- **For Loop:** Iterates over a list of items.
- **While Loop:** Repeats as long as a condition is true.
- **Until Loop:** Repeats until a condition becomes true.

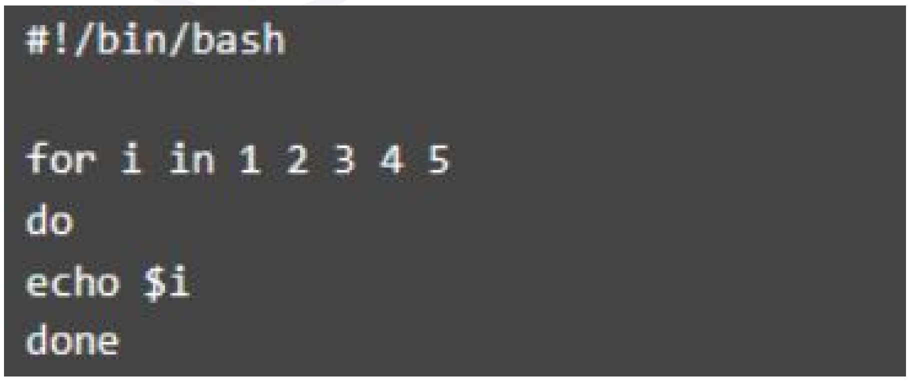
For Loop
A for loop iterates over a list of values or a range.

Syntax:



```
for variable in list
do
statement
done
```

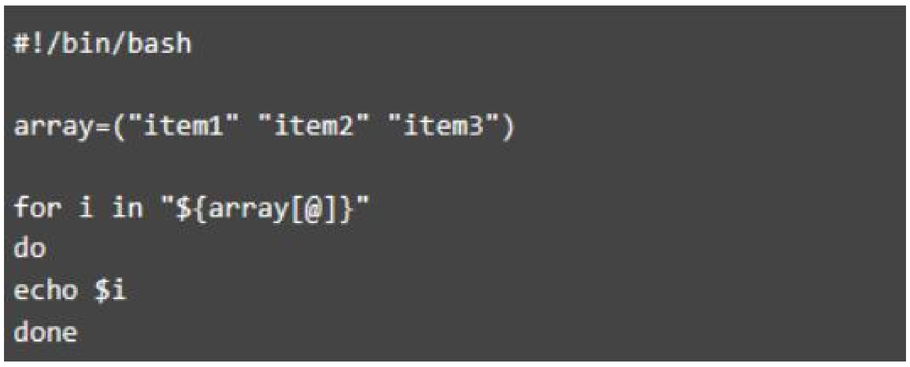
Example: Prints the numbers from 1 to 5:



```
#!/bin/bash

for i in 1 2 3 4 5
do
echo $i
done
```

In this example, the script iterates through the numbers from 1 to 5 and prints each number using the echo command.

Example: To iterate through the items in an array:



```
#!/bin/bash

array=("item1" "item2" "item3")

for i in "${array[@]}"
do
echo $i
done
```

In this example, the script iterates through the items in the array and prints each item using the echo command.

While Loop
A while loop repeats as long as a specified condition is true.

Syntax:

```
while [ condition ]; do
  # Code to execute while the condition is true
done
```

In this example, the script iterates through the items in the array and prints each item using the echo command.

While Loop
A while loop repeats as long as a specified condition is true.

Syntax:

```
while [ condition ]; do
  # Code to execute while the condition is true
done
```

Example: Counting down from 5

```
#!/bin/bash

counter=5

while [ $counter -gt 0 ]; do
  echo "Counter: $counter"
  counter=$((counter - 1))
done
```

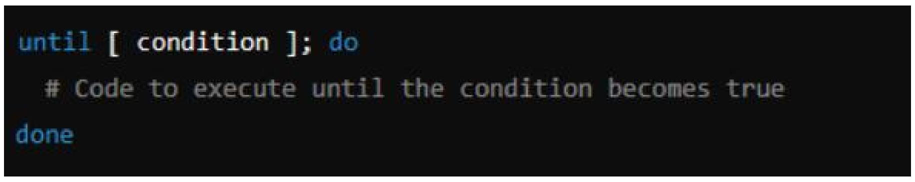
Example: To read lines from a file

```
#!/bin/bash

while read line
do
echo $line
done < /home/fosslinux/Documents/myparameters.txt
```

In this example, the script reads each line from the myparameters.txt file using the read command and assigns it to the line variable. It then prints each line using the echo command.

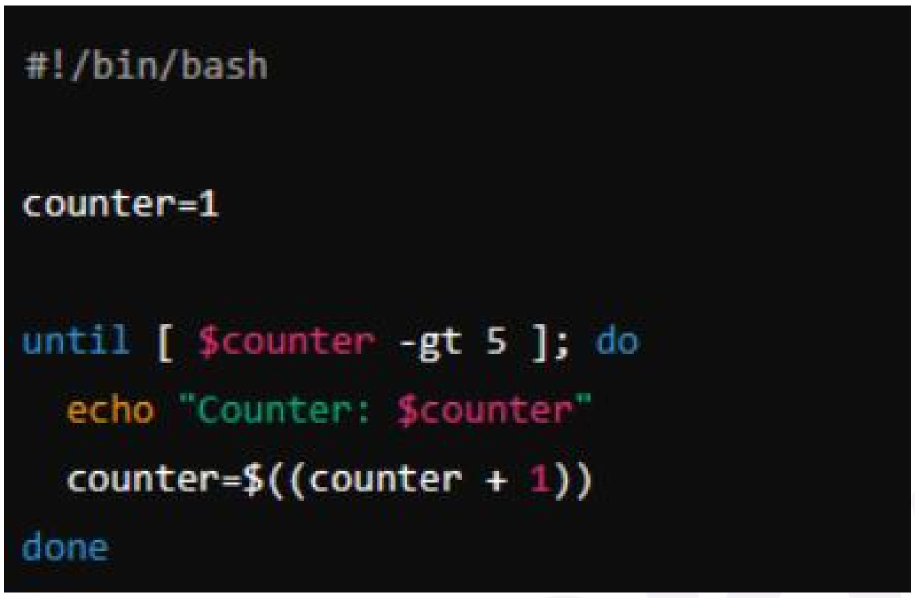
Until Loop
An until loop is the opposite of a while loop. It repeats until a condition becomes true.

Syntax:



```
until [ condition ]; do
  # Code to execute until the condition becomes true
done
```

Example: Counting up to 5



```
#!/bin/bash

counter=1

until [ $counter -gt 5 ]; do
  echo "Counter: $counter"
  counter=$((counter + 1))
done
```