UTSA

The University of Texas at San Antonio

University Technology Solutions

Express Bash Scripting Tutorial Part 2

Quickly Learn Bash Scripting in Linux

Brent League

Overview

- The If Statement
- The Else Clause
- The Elif Clause
- The While Loop
- The For Loop
- Using Break and Continue

The If Statement

Create a new script called `if.sh` and make it executable as shown in part one of our Bash training sessions.

```
COLOR=$1
if [[ $COLOR = “blue" ]]
then
  Echo "The color is blue"
fi

USER_GUESS=$2
#Set Computer’s value to 50
COMPUTER=50

if [[ $USER_GUESS -lt $COMPUTER ]]
then
  echo "You're too low"
fi
```

Boolean Values

We used the less than (or –lt) value in the previous script, but we have a lot more options using Boolean values when comparing numbers. For example:

- -eg – if equal
- -ne – if not equal
- -lt – if less than
- -gt – if greater than
- -le – if less than or equal
- -ge – if greater than or equal

The If Statement

Create a new script called `if.sh` and make it executable by typing: `chmod +x if.sh`

```
COLOR=$1
if [[ $COLOR = "blue" ]]
then
  Echo "The color is blue"
fi

USER_GUESS=$2
#Set Computer's value to 50
COMPUTER=50

if [[ $USER_GUESS -lt $COMPUTER ]]
then
  echo "You're too low"
fi
```

The Else Clause

- The “else” clause allows us to perform one task if the expression is true, and perform a different task if the expression is false
- If the expression is false, the commands following the “else” command up to the “fi” command are executed
- If the expression is true, the script will execute the commands between “then” and “fi”

Using the Else Clause in a Script

```
#!/usr/bin/env bash
COLOR=$1
if [[ $COLOR = "blue" ]]
then
 echo "The color is blue"
else
 echo "The color is NOT blue"
fi

USER_GUESS=$2
COMPUTER=50

if [[ $USER_GUESS -lt $COMPUTER ]]
then
 echo "You're too low"
else
 echo "You're equal or too high"
fi
```

The Elif Clause

- The “elif” clause stands for “else if”
- It allows us to check for a different expression than the one used in the “if”
- “elif” must come before the “else” clause which must be the last clause in the “if” statement
- Let’s add the following to our `if.sh` script right above the else clause (which must be the

```
elif [[ $USER_GUESS -gt  $COMPUTER ]]
then
 echo "You're too high"
else
 echo "You've guessed it"
```

The While Loop

- Loops give us the ability to execute our code repetitively
- Let's create a script and call it `while.sh`

```
#!/usr/bin/env bash

COUNT=0

while [[ $COUNT -lt 10 ]]
do
 echo "COUNT = $COUNT"
 ((COUNT++))
done

echo "while loop finished"
exit 0
```

The For Loop

- The “for” statement used in conjunction with the loop command is used to instruct our script to perform a function that is followed by our “for statement.
- In previous exercises, we’ve asked for parameters individually by using $1, $2, etc. But this time, we’ll use a special symbol that is entered as $@
- The $@ symbol holds all of the values a user enters in one array.

The For Loop

Let's create a new script called `for.sh` and include the following text (:

```
#!/usr/bin/env bash

NAMES=$@
for NAME in $NAMES
do
 echo "Hello $NAME"
done
echo "for loop terminated"
exit 0
```

The For Loop

Now run the command and enter a name, or multiple names separated by a space:

```
./for.sh Brent Aurin Tina Bob
```

The output should look like this:

```
Hello Brent
Hello Aurin
Hello Tina
Hello Bob
for loop terminated
```

Using Break with Loops

- There are two special instructions that can be used with loops
- Let's talk about the break instruction first
- Break causes the current loop to terminate if a certain value is provide by a user and it will then begin executing any instructions AFTER the done statement in your script
- Let's take a look at it's function in the next slide

Using Break with Loops

```
#!/usr/bin/env bash
NAMES=$@

for NAME in $NAMES
do
 if [[ $NAME = "Sally" ]]
 then
  break
 fi
 echo "Hello $NAME"
done

echo "for loop terminated"
exit 0
```

Using Continue with Loops

- In our previous example, you can see that the break instruction went to the end of the loop
- In contrast to the break instruction, the continue instruction goes to the top of the loop
- Let's edit the `for.sh` script again
- In this case, we are simply going to replace the break instruction with the continue instruction and see what happens

Using Continue with Loops

```
#!/usr/bin/env bash

NAMES=$@

for NAME in $NAMES
do
 if [[ $NAME = "Sally" ]]
 then
  continue
 fi
 echo "Hello $NAME"
done

echo "for loop terminated"
exit 0
```

UTSA

The University of Texas at San Antonio

University Technology Solutions

Express Bash Scripting Tutorial Part 2

Quickly Learn Bash Scripting in Linux

Brent League