ITE315 Module 2 Part B - Bash As A Programming Language: Repetition Statements

Athens State University

Contents

1 The Wheel Goes Around

Loops in Bash: The `while` Loop

- The `while` loop allows for the repetitive execution of a list of commands so long as the command controlling the loop returns an exit status of zero

```
while CONTROL-COMMAND ; do
   LISTOFCOMMANDS;
done
```

A Simple `while` Loop

```
#!/bin/bash
counter=1
while [ $counter -le 10]; do
   echo $counter
   (( counter++ ))
done
```

Pretty simple, no? Works much the same as what we've been using to dealing with in C++ once you get past working with commands rather than Boolean conditionals.

The `until` Loop

```
#/bin/bash
counter=1
unitl [ $counter -gt 10 ]
do
   echo $counter
   (( counter++ ))
done
```

The `until` loop has similar syntax as does the `while` loop but the difference is that the loop will continue until the test becomes true. Note how the test changed: we reversed the direction of the inequality. Thus this loop does exactly the same thing as the example `while` loop.

2 Loops in Bash: The `for` Loop

Loops in Bash: The `for` Loop

- The `for` loop in Bash is different than in other programming languages
  - You use to iterate over a string of 'words' within a string
  - For other data types, you will want a command that outputs a list of strings

```
for $var in LIST ; do
   COMMANDS
done
```

A Simple `for` Loop

```
#!\bin\bash
names='Stan Kyle Cartman'
for name in $name ; do
    echo $name
done
echo Done listing $names
```

This script creates a simple list of names. For each of the items in the list, assign to the variable and do the following commands: echo the name to the screen. We can have as many commands as needed between the `do` and `done` keywords.

Ranges

```
#!/bin/bash
for value in {1..5}; do
   echo $value
done
```

- Make certain that you leave no spaces between the braces. If you do, then Bash interprets that as a list of items rather than a range
- If the first value is larger than the second, then loop counts downward.

It's also possible to specify an increment or decrement value:

```
#!/bin/bash
for value in {10..0..2}
do
   echo $value
done
```

In this loop, the variable will be decremented by 2 on each pass through the loop.

A Good Example Of The Power Of Loops

```
#!/bin/bash
for value in $1/*.html
do
  cp $value $1/$(basename -s .html $value).php
done
```

This code will copy all files in a folder whose name has the ".html" extension to new files with the ".php" extension.