Shell Scripting – Part III

Loops

- Various Loops exist in Bash
  - while
  - until
  - for
  - select

While Loops

- Syntax

```
while condition
do
    … Code to iterate …
done
```

- Example

```
x=0
while [ $x -lt 10 ]
do
    echo $x
    x=$(( $x + 1 ))
done
```

Until Loops

- Syntax

```
until condition
do
    … Code to iterate …
done
```

- Example

```
x=0
until [ $x -ge 10 ]
do
    echo $x
    x=$(( $x + 1 ))
done
```

For Loops

- Syntax

```
for loop_var in list
do
    … Code to iterate …
done
```

- Example

```
names=`Luke Leia Han Anakin ObiWan’
for value in $names
do
    echo $value
done
```

For Loops (continued)

- Really useful for iterating through command line arguments
- Try the following:
- Example

```
for value in $@
do
    echo $value
done
```

- ./script.sh Arg1 Arg2 Arg3

For Loops (continued)

- Now Try the following:
- Example

```
for value in $*
do
      echo $value
done
```

- ./script.sh Arg1 Arg2 Arg3

For Loops (continued)

- Now Try the following:
- Example

```
for value in $*
do
      echo $value
done
```

- ./script.sh Arg1 Arg2 Arg3 “Arg4 Arg4”

For Loops (continued)

- Now Try the following:
- Example

```
for value in $@
do
        echo $value
done
```

- ./script.sh Arg1 Arg2 Arg3 “Arg4 Arg4”

For Loops (continued)

- Now Try the following:
- Example

```
for value in “$*”
do
      echo $value
done
```

- ./script.sh Arg1 Arg2 Arg3 “Arg4 Arg4”

For Loops (continued)

- Now Try the following:
- Example

```
for value in “$@”
do
      echo $value
done
```

- ./script.sh Arg1 Arg2 Arg3 “Arg4 Arg4”

For Loops (continued)

- Let us discuss this behavior

For Loops (continued)

- Helpful list creators exist
- Try

```
for value in {1..5}
do
    echo $value
done
```

- The general syntax is {start..end..step}
- It can even step backwards if start > end

Break and Continue

- The Keywords break and continue exist for all loops
- Behave exactly the same as other languages

Reading Files

- Loops can be used to read files line-by-line

```
input="/path/to/txt/file"

while IFS= read -r line
do
    echo "$line"
done < "$input"
```

Understand what IFS (Internal Field Separator), -r, read, and < (Input redirection) do.