BASH SCRIPTING

Loops

CONTENTS

While Loop

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

Until Loop

```bash
#!/bin/bash
# Basic until loop
counter=1
until [ $counter -gt 10 ]
do
echo $counter
((counter++))
done
echo All done
```

For Loop

```
#!/bin/bash
# Basic for loop
names='Stan Kyle Cartman'
for name in $names
do
echo $name
done
echo All done
```

For with Range

```bash
#!/bin/bash
# Basic range in for loop
for value in {1..5}
do
echo $value
done
echo All done
```

For Loop Stepping

It is also possible to specify a value to increase or decrease by each time. You do this by adding another two dots (..) and the value to step by.

```bash
#!/bin/bash
# Basic range with steps for loop

for value in {10..0..2}
do
echo $value
done

echo All done
```

For loop to process set of Files

One of the more useful applications of **for** loops is in the processing of a set of files. To do this we may use wildcards. Let's say we want to convert a series of .html files over to .php files.

```
#!/bin/bash
# Make a php copy of any html files
for value in $1/*.html
do
cp $value $1/$( basename -s .html $value ).php
done
```

Break

The **break** statement tells Bash to leave the loop straight away. It may be that there is a normal situation that should cause the loop to end but there are also exceptional situations in which it should end as well. For instance, maybe we are copying files but if the free disk space get's below a certain level we should stop copying.

```bash
#!/bin/bash
# Make a backup set of files
for value in $1/*
do
used=$( df $1 | tail -1 | awk '{ print $5 }' | sed 's/%//' )
if [ $used -gt 90 ]
then
echo Low disk space 1>&2
break
fi
cp $value $1/backup/
done
```

Continue

The **continue** statement tells Bash to stop running through this iteration of the loop and begin the next iteration. Sometimes there are circumstances that stop us from going any further. For instance, maybe we are using the loop to process a series of files but if we happen upon a file which we don't have the read permission for we should not try to process it.

```
#!/bin/bash
# Make a backup set of files
for value in $1/*
do
if [ ! -r $value ]
then
echo $value not readable 1>&2
continue
fi
cp $value $1/backup/
done
```

Menu using Select

```bash
#!/bin/bash
# A simple menu system
names='Kyle Cartman Stan Quit'
PS3='Select character: '
select name in $names
do
if [ $name == 'Quit' ]
then
break
fi
echo Hello $name
done
echo Bye
```

Difference between $a and $* and install multiple packages using command line arguments.

```
#!/bin/bash
sudo apt update -y
for app in "$@"; do
  echo "Installing $app..."
  sudo apt install -y "$app"
done
```

- **$@** takes all the command line arguments as an array while **$*** differentiates the command line using words.

- The file names cli_arg.sh can be executed with command line arguments as:

```
$./cli_arg.sh wget nginx
```

- This will install wget nginx using for loop.