Linux/Unix System Programming

CSCI 2153

David L. Sylvester, Sr., Professor

BASH Programming

Loops:

The **for** loop is a little bit different from other programming languages. Basically, it let's you iterate over a series of 'words' within a string.

```
#!/bin/bash
clear
echo
echo
echo Enter your favorite phrase
read phrase
for i in $phrase; do
        echo
        echo item: $i
done
echo
echo
for e in $( ls ); do
        echo item: $e
done
echo
echo
```

On the seventh line, we declare i to be the variable that will take the different values contained in $phrase
The eigth line could be longer if needed, or there could be more lines before the **done**.
'**done**' indicates that the code that used the value of $i has finished and $i can take a new value.
This script has very little sense, but a more useful way to use the for loop would be to use it to match only certain files.

BASH Programming

Loops:

The **while** executes a piece of code if the control expression is true, and only stops when it is false (or a explicit break is found within the executed code.

```
#!/bin/bash
clear
echo
echo
counter=0;
while [ $counter -lt 10 ]; do
        echo The counter is $counter
        let counter=counter+1;
done
echo
echo
```

This loop sets the variable **counter** to ‘0’. Then performs the while loop starting counter at ‘0’, looping while **counter** is less than ‘10’.

Each time the loop is performed, it displays The counter is, and the value of **counter**. Then increments **counter** by ‘1’ before returning to the beginning of the loop.

BASH Programming

Loops:

The **until** loop is almost equal to the while loop, except that the code is executed while the control expression evaluates to false.

```
#!/bin/bash
clear
echo
echo
counter=10;
until [ $counter -lt 0 ]; do
        echo The counter is $counter
        let counter-=1;
done
echo
echo
~
```

This loop sets the variable **counter** to ‘10’. Then performs the while loop starting counter at ‘10’, looping until **counter** is less than ‘0’.

Each time the loop is performed, it displays “The counter is”, and the value of **counter**. Then decrements **counter** by ‘1’ before returning to the beginning of the loop.

BASH Programming

Functions:

As in almost any programming language, you can use functions to group pieces of code in a more logical way or practice the divine art of recursion. Declaring a function is just a matter of writing function my_func { my_code }.Calling a function is just like calling another program, you just write its name.

```
#!/bin/bash
function header {
        echo
        echo Linux/Unix Programming
        echo Baton Rouge Community College
        echo Fall 2020
        echo
}
function footer {
        echo
        echo _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
        echo Program has endded.
        echo - - - - - - - - - - - - - - - - - - - -
}
clear
header
counter=10;
until [ $counter -lt 0 ]; do
        echo The counter is $counter
        let counter-=1;
done
footer
```

Defining **header** function

Defining **footer** function

Calling **header** function

Calling **footer** function

BASH Programming

Using select to make simple menus:

Notice that it's very similar to the 'for' construction, only rather than looping for each 'word' in $OPTIONS, it prompts the user.

```
#!/bin/bash
OPTIONS="Hello Quit"
select opt in $OPTIONS; do
        if [ "$opt" = "Quit" ]; then
                echo Done...
                exit
        elif [ "$opt" = "Hello" ]; then
                echo Hello world...
                exit
        else
                clear
                echo BAD OPTION...
        fi
done
```