BASH Programming – Introduction

1. Very simple Scripts

- Traditional hello world script:

```
#!/bin/bash
echo Hello World
```

- A very simple backup script:

```
#!/bin/bash
tar -cZf /var/my-backup.tgz /home/me/
```

2. Redirection

- stdout 2 file:

```
ls -l > ls-l.txt
```

- stderr 2 file:

```
grep da * 2> grep-errors.txt
```

- stdout 2 stderr:

```
grep da * 1>&2
```

- stderr 2 stdout:

```
grep * 2>&1
```

- stderr and stdout 2 file:

```
rm -f $(find / -name core) &> /dev/null
```

3. Pipes

- let you use the output of a program as the input of another one
- simple pipe with sed:

```
ls -l | sed -e "s/[aeio]/u/g"
```

- an alternative to ls -l *.txt:

```
ls -l | grep "\.txt$"
```

4. Variables

- You can use variables as in any programming languages.
- There are no data types.
- A variable in bash can contain a number, a character, a string of characters.
- You have no need to declare a variable, just assigning a value to its reference will create it.

4.1 Hello World! using variables

```
#!/bin/bash

STR="Hello World!"

echo $STR
```

- if you don't use the '$' sign, the output of the program will be different

4.2 A very simple backup script

```
#!/bin/bash
OF=/var/my-backup-$(date +%Y%m%d).tgz
tar -cZf $OF /home/me/
```

4.3 Local variables

```
#!/bin/bash
                HELLO=Hello
                function hello {
                        local HELLO=World
                        echo $HELLO
                }
                echo $HELLO
                hello
                echo $HELLO
```

5. Conditionals

if [expression];

then

code if 'expression' is true.

fi

5.1 Basic conditional: if .. then

```
#!/bin/bash
            if [ "foo" = "foo" ]; then
               echo expression evaluated as true
            fi
```

5.2 Basic conditional if .. then ... else

```
#!/bin/bash
            if [ "foo" = "foo" ]; then
               echo expression evaluated as true
            else
               echo expression evaluated as false
            fi
```

5.3 Conditionals with variables

```
#!/bin/bash
            T1="foo"
            T2="bar"
            if [ "$T1" = "$T2" ]; then
                echo expression evaluated as true
            else
                echo expression evaluated as false
            fi
```

6. Loops for, while and until

- The **`for`** loop: let's you iterate over a series of 'words' within a string.
- 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.)
- The **`until`** loop is almost equal to the while loop, except that the code is executed while the control expression evaluates to false.

6.1 For sample

```
#!/bin/bash
        for i in $( ls ); do
            echo item: $i
        done
```

6.2 C-like for

```
#!/bin/bash
        for i in `seq 1 10`;
        do
                echo $i
        done
```

6.3 While sample

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

6.4 Until sample

```
#!/bin/bash
        COUNTER=20
        until [  $COUNTER -lt 10 ]; do
            echo COUNTER $COUNTER
            let COUNTER-=1
        done
```

7. Functions

- 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.

7.1 Functions sample

```
#!/bin/bash
            function quit {
                exit
            }
            function hello {
                echo Hello!
            }
            hello
            quit
            echo foo
```

7.3 Functions with parameters

```
#!/bin/bash
                function quit {
                   exit
                }
                function e {
                    echo $1
                }
                e Hello
                e World
                quit
                echo foo
```

8.1 User interfaces - Menu

```
#!/bin/bash
            OPTIONS="Hello Quit"
            select opt in $OPTIONS; do
                if [ "$opt" = "Quit" ]; then
                 echo done
                 exit
                elif [ "$opt" = "Hello" ]; then
                 echo Hello World
                else
                 clear
                 echo bad option
                fi
            done
```

8.2 Using the command line

```
#!/bin/bash
        if [ -z "$1" ]; then
            echo usage: $0 directory
            exit
        fi
        SRCD=$1
        TGTD="/var/backups/"
        OF=home-$(date +%Y%m%d).tgz
        tar -cZf $TGTD$OF $SRCD
```

9.1 Misc - Reading user input

```
 #!/bin/bash
        echo Please, enter your name
        read NAME
        echo "Hi $NAME!"

#!/bin/bash
        echo Please, enter your firstname and lastname
        read FN LN
        echo "Hi! $LN, $FN !"
```

9.2 Arithmetic evaluation

```
echo 1 + 1
echo $((1+1))
echo $[1+1]
```

9.3 Getting the return value

```
#!/bin/bash
        cd /dada &> /dev/null
        echo rv: $?
        cd $(pwd) &> /dev/null
        echo rv: $?
```

9.4 Capturing a commands output

```
#!/bin/bash
        DBS=`mysql -uroot  -e"show databases"`
        for b in $DBS ;
        do
                mysql -uroot -e"show tables from $b"
        done
```

10.1 String comparison operators

```
(1) s1 = s2
(2) s1 != s2
(3) s1 < s2
(4) s1 > s2
(5) -n s1
(6) -z s1
```

10.2 String comparison examples

```
#!/bin/bash
        S1='string'
        S2='String'
        if [ $S1!=$S2 ];
        then
                echo "S1('$S1') is not equal to S2('$S2')"
        fi
        if [ $S1=$S1 ];
        then
                echo "S1('$S1') is equal to S1('$S1')"
        fi
```

10.3 Arithmetic operators

- +
- -
- *
- /
- % (remainder)
- -lt (<)
- -gt (>)
- -le (<=)
- -ge (>=)
- -eq (==)
- -ne (!=)

11.1 A very simple backup script

```
#!/bin/bash
            SRCD="/home/"
            TGTD="/var/backups/"
            OF=home-$(date +%Y%m%d).tgz
            tar -cZf $TGTD$OF $SRCD
```

11.2 File renamer (simple)

```
#!/bin/bash
     # renames.sh
     # basic file renamer

     criteria=$1
     re_match=$2
     replace=$3

     for i in $( ls *$criteria* );
     do
         src=$i
         tgt=$(echo $i | sed -e "s/$re_match/$replace/")
         mv $src $tgt
     done
```