Variables

- variable is simply a place where we can store and retrieve assigned information from.
- When referencing a variable you must always precede the variable name with a "$" dollar sign.

Creating Variables

```
john@john-desktop:~$ name=john
john@john-desktop:~$ age=21
john@john-desktop:~$ echo $name
john
john@john-desktop:~$ echo $age
21
john@john-desktop:~$ echo "My name is $name and I am $age years old"
My name is john and I am 21 years old
```

Read Command

command accepts a line of input and spits it into variables.

Each argument that is read must be the name of a variable.

john@john-desktop:~/scripts$ read firstname surname telephone
john smith 0123456789
john@john-desktop:~/scripts$ echo $firstname
john
john@john-desktop:~/scripts$ echo $surname
smith
john@john-desktop:~/scripts$ echo $telephone
0123456789

IFS - Internal Field Separator

To change the delimiter used by the read command, we need to modify the value held within the "IFS" Internal Field Separator variable.

Note, it is good practise to set the IFS back to its default setting after you have processed your line with the read command.

```
john@john-desktop:~/scripts$ old_IFS=$IFS
john@john-desktop:~/scripts$ IFS=-
john@john-desktop:~/scripts$ read day month year
25-12-2013
john@john-desktop:~/scripts$ echo $day
25
john@john-desktop:~/scripts$ echo $month
12
john@john-desktop:~/scripts$ echo $year
2013
john@john-desktop:~/scripts$ IFS=old_IFS
```

Some basic commands

env command
The env command can be used to display only exported variables.

set command

The set command is used to display all variable both local and exported variables.

printf command

printf command controls the output of data similar to that of the "printf" command within the "C"

programming language"

john@john-desktop:~/scripts$ name=john
john@john-desktop:~/scripts$ day=monday
john@john-desktop:~/scripts$ printf "My name is $name\nToday is $day\n"
My name is john
Today is monday

positional parameters

vim script01.sh

```
#!/bin/bash
printf "\nName of script $0\n"
printf "Parameters Passed: $1 $2 $3 $4 $5 $6\n"
```

```
john@john-desktop:~/scripts$ ./script01.sh first second third fourth fifth sixth

Name of script ./script01.sh
Parameters Passed: first second third fourth fifth sixth
```

declare statement

"declare" statement is used to modify the properties of a variable

john@john-desktop:~/scripts$ MyString="Land of Linux"
john@john-desktop:~/scripts$ declare -p MyString
declare -- MyString="Land of Linux"

Declaration Type
-a Variable is an Array
-f Use function names only
-i Variable is an integer
-p Used to display attributes of variable
-r read only - Makes variables read-only

john@john-desktop:~/scripts$ declare -r rovariable=2468

john@john-desktop:~/scripts$ declare -p rovariable

declare -r rovariable="2468"
john@john-desktop:~/scripts$ rovariable=1357
bash: rovariable: readonly variable