The while loop syntax

The syntax is:

```
          while [ condition ]
          do
                command1
                command2
                ..
                ....
                commandN
          done
```

Command1..commandN will execute while a condition is true. To read a text file line-by-line, use the following syntax:

```
          while IFS= read -r line
          do
                command1 on $line
                command2 on $line
                ..
                ....
                commandN
          done < "/path/to/filename"
```

OR

```
          while IFS= read -r field1 filed2 field3 ... fieldN
          do
                command1 on $field1
                command2 on $field1 and $field3
                ..
                ....
                commandN on $field1 ... $fieldN
          done < "/path/to dir/file name with space"
```

IFS is used to set field separator (default is while space). The -r option to read command disables backslash escaping (e.g., \n, \t). This is failsafe while read loop for reading text files.

while loop Example

Create a shell script called while.sh:

```
#!/bin/bash
# set n to 1
n=1

# continue until $n equals 5
while [ $n -le 5 ]
do
        echo "Welcome $n times."
        n=$(( n+1 ))     # increments $n
done
```

Save and close the file. Run it as follows:

```
chmod +x while.sh
./while.sh
```

Sample outputs:

```
Welcome 1 times.
Welcome 2 times.
Welcome 3 times.
Welcome 4 times.
Welcome 5 times.
```

The script initializes the variable n to 1, and then increments it by one. The while loop prints out the "Welcome $n times" until it equals 5 and exit the loop.

Using ((expression)) Format With The While Loop

You can use ((expression)) syntax to test arithmetic evaluation (condition). If the value of the expression is non-zero, the return status is 0; otherwise the return status is 1. To replace while loop condition **while [$n -le 5]** with **while ((num <= 10))** to improve code readability:

```
#!/bin/bash
n=1
while (( $n <= 5 ))
do
        echo "Welcome $n times."
        n=$(( n+1 ))
done
```

Reading A Text File

You can read a text file using read command and while loop as follows (whilereadfile.sh):

```
#!/bin/bash
file=/etc/resolv.conf
while IFS= read -r line
do
         # echo line is stored in $line
        echo $line
done < "$file"
```

Save and close the file. Run it as follows:

```
chmod +x whilereadfile.sh
./whilereadfile.sh
```

Sample outputs:

```
nameserver 127.0.0.1
nameserver 192.168.1.254
nameserver 4.2.2.1
```

Reading A Text File With Separate Fields

You can store above output in two separate fields as follows (whilereadfields.sh):

```
#!/bin/bash
file=/etc/resolv.conf
# set field separator to a single white space
while IFS=' ' read -r f1 f2
do
        echo "field # 1 : $f1 ==> field #2 : $f2"
done < "$file"
```

Run it as follows:

```
chmod +x whilereadfields.sh
./whilereadfields.sh
```

Sample outputs:

```
field # 1 : nameserver ==> field #2 : 127.0.0.1
field # 1 : nameserver ==> field #2 : 192.168.1.254
field # 1 : nameserver ==> field #2 : 4.2.2.1
```

Another useful example for reading and phrasing /etc/passwd file using the while loop (readpasswd.sh):

```
#!/bin/bash
file=/etc/passwd
# set field delimiter to :
# read all 7 fields into 7 vars
while IFS=: read -r user enpass uid gid desc home shell
do
    # only display if UID >= 500
        [ $uid -ge 500 ] && echo "User $user ($uid) assigned \"$home\" home
directory with $shell shell."
done < "$file"
```

Save and close the file. Run it as follows:

```
chmod +x readpasswd.sh
./readpasswd.sh
```

Sample output:

```
User nobody (65534) assigned "/nonexistent" home directory with /bin/sh shell.
User vivek (1000) assigned "/home/vivek" home directory with /bin/bash shell.
User oracle (1004) assigned "/usr/lib/oracle/xe" home directory with /bin/bash shell.
User simran (1001) assigned "/home/simran" home directory with /bin/bash shell.
User t2 (1002) assigned "/home/t2" home directory with /usr/local/bin/t2.bot shell.
```