**Loops and aliases**

**for** and **while** oops in bash can be useful for repeating tasks. They are commands that can be used in a number of contexts, including by grep and sed.

**1\_for** loops iterate through a set of values until the list is exhausted:

**for** VARIABLE **in** 1 2 3 4 5 .. N

**do**

command1

command2

commandN

**done**

For example:

for i in hello 1 "\*" 2 goodbye

do

echo "Looping ... i is set to $i"

done

for i in 1 2 3 4 5

do

echo "Looping ... number $i"

done

#You can define series….

for i in `seq 1 10`; do echo $i; done

for variable in {1..10}; do echo $variable; done

Sometimes you will see for loops on one line:

for i in hello 1 "\*" 2 goodbye; do echo "Looping ... i is set to $i"; done

#Note the semicolons that separate commands.

**2\_for** can also be used to select files

**for** VARIABLE **in** file1 file2 file3

**do**

command1 on $VARIABLE

command2

commandN

**done**

for FILE in t\*

do

echo "xx " $FILE

grep "Superman" $FILE >>outfile

done

#will print out all files that begin with t and extract all lines of Superman.

**3\_for** can also take command output.

for OUTPUT in $(Linux-Or-Unix-Command-Here)

do

command1 on $OUTPUT

command2 on $OUTPUT

commandN

done

For example:

for i in $(ls \*.txt) ; do echo "The directory has these text files: $i"; echo "Thank you"; done

Here is an example where you have many sequence motifs in a file and want to see the number of times each one matches a reference sequence. Your sequence motifs in one file are on different lines, and you want the output tab delimited.

for i in $(cat temp2); do echo $i | tr '\n' '\t'; grep -c $i file\_2.fa; done > outfile

#file1 has the motifs (temp2)

ATGCAT

TGCCTA

CGTATT

## **while Loops**

**while** loops say, “While an expression is true, keep executing these lines of code.” The loop has the following format:

**while** [ <some test> ]

**do**

<commands>

**done**

count="0"

max="10"

while [ $count != $max ]; do count=`expr $count + 1`

echo "We are now at number: $count"

done

Here is a use of while that matches two different columns in different files.

cut -f 1 file1.txt | while read i ; do grep -A 1 $i file2.txt; done > outfile

file1.txt

element1 40 45 attributes

element2 120 125 attributes

element3 140 145 attributes

file2.txt

element1

AGTCT

element2

GGGAA

element3

TTTAA

element4

TTAAA

element5

GGTTT

**Introduction to aliases**

You can create a new name for an existing command. It could make your life easier

For example:

alias ls='ls -p'

#now when one types ls, it executes ls -p

\ backslash returns normal behaviour.

or between ' '; e.g. \ls or 'ls'

If you are making your own command, then you want to check you are not overwriting another. For example, you want to make “fg” a new name for a command. Does the “fg” command already exist? The which command will tell you if a command already exists. It searches your $PATH to find commands and the path to them.

which fg

/usr/bin/fg

so fg does exist!

How about calling the alias “feelgood?”

which feelgood

This command returns nothing, Let’s use it.

alias feelgood= 'echo "Feeling great!" '

feelgood

What happens? Try the alias command. Aliases will last only as long as your session.