**Filters**

* cat count.txt tac count.txt: Reverse of cat function
* tee: tee is almost the same as cat, except that it has two identical outputs.

tac count.txt | tee tempt.txt | tac

* grep:

cat tennis.txt | grep Williams

Serena Williams, USA

Venus Williams, USA

* cut: The cut filter can select columns from files, depending on a delimiter or a count of bytes.

-d means delimiter, colon (:) is used as a delimiter, f means field.

aaron:x:1001:1001:aaron,,,:/home/aaron:/bin/bash

james:x:1005:1009:james,,,:/home/james:/bin/bash

walter:x:1006:1006:walter clarus:/home/walter:/bin/sh

cut -d: -f1-3 /etc/passwd | tail -3

aaron:x:1001

james:x:1005

walter:x:1006

* tr: It is used for translating and deleting characters.

cat clarusway.txt | tr -d e

way to rinvnt yourslf

cat clarusway.txt | tr [a-z] [A-Z]

WAY TO REINVENT YOURSELF

cat clarusway.txt | tr [:space:] '\t'

WAY TO REINVENT YOURSELF

* wc: Counting lines, words and characters.

wc tennis.txt

5 15 100 tennis.txt 🡺 5 lines, 15 words, 100 bits(chars)

wc -l tennis.txt 🡪 5

wc -w tennis.txt 🡪 15

wc -c tennis.txt 🡪 100

wc -m tennis.txt 🡪 characters

* sort: The sort filter will default to an alphabetical sort.

sort music.txt

Abba

Brel

Led Zeppelin

sort -r the flag returns the results in reverse order

sort -f the flag does case insensitive sorting

sort -n the flag returns the results as per numerical order

* uniq: With uniq you can remove duplicates from a sorted list.

sort music.txt |uniq

Abba

Brel

Queen \*There was another Queen

* comm: Comparing streams (or files)
* column 1 lines unique to list1
* column 2 lines unique to list2
* column 3 lines that appear in both files

comm list1.txt list2.txt

* Abba
* Bowie
* Cure
* Queen
* Sweet
* Turner

## Control Operators

## ; More than one command can be used in a single line.

## & Command ends with & and doesn't wait for the command to finish.

## $? Command is used to check the status of last executed command

## && The shell will interpret && as a logical AND. When using && the second command is executed only if the first one succeeds (returns a zero exit status).

## || The || represents a logical OR. The second command is executed only when the first command fails (returns a non-zero exit status).

## # Comment sign. Anything was written after # will be ignored.

## && and || Use this logical AND and logical OR to write an if-then-else structure on the command line. This example uses echo to display whether the rm command was successful.

## rm file1 && echo It worked! || echo It failed!

## It worked!

## \ Escape characters are used to remove the special meaning from a single character. A non-quoted backslash, \, is used as an escape character in Bash. It preserves the literal value of the next character that follows, with the exception of newline.