Introduction to Bash video lecture

13 - Pipelines - combining commands using pipes

Using pipes

- A pipe | makes the output from one command the input to the next
- Makes the Bash text commands a veritable toolbox for doing almost anything with just one command line
- Saves a lot of time (no need to write a program)
- Is the one thing you should learn to get the most from your shell

Examples

```
$ grep -iw egg meals.txt | wc -l  # how many meals with egg?
2
$ cut -d ':' -f 2- < meals.txt | sort
 Apple
 Egg and coffee
 Egg and tea
 Fish and chips
 Fish and potato
 Hamburger and coke
 Pasta and wine
 Peanuts and beer
 Pizza
 Sandwich and juice
 Sandwich and milk
 Stake and sallad
```

Examples

```
$ grep Breakfast meals.txt | cut -d ':' -f2- | sort
 Egg and coffee
 Egg and tea
 Sandwich and milk

$ # Let’s remove the leading blanks!
$ grep Breakfast meals.txt | cut -d ':' -f2- | sort | cut -d ' ' -f2-
Egg and coffee
Egg and tea
Sandwich and milk
```

Getting the word frequency from a text file

```
$ tr ' ' '\n' < lorem80.txt | tr -d '[.,:;!?]'| grep -v '^$'| sort -i | uniq -ic | sort -rnk1 | head -11
     11 sed
      8 in
      8 dolor
      8 at
      6 vel
      6 id
      6 elit
      6 ac
      5 tortor
      5 quis
      5 Donec
```

Getting the word frequency from a text file

```
$ tr ' ' '\n' < lorem80.txt |
# make all spaces into newlines (so that we get one word per line)
  tr -d '[.,:;!?]' |
# delete all punctuation listed inside the square brackets
  grep -v '^$' |
# find all lines that are not empty
  sort -i |
# sort the result caseinsensitively
  uniq -ic |
# remove duplicate lines and report the number of dupes
  sort -rnk1 |
# sort the result descendingly, using first column, treated numerically
  head -11
# keep only the eleven first lines (the top 11 frequencies)
```

<<last page>>