Pipes Exercises (1)

The command `ls | head` runs ls and head and pipes the standard output of ls into the standard input of head.

The following shell commands are particularly useful in pipes:

- `cat [FILENAME [FILENAME...]]` writes the contents of one or more files to standard output. This is a good way of starting a pipe. If you leave off all the filenames, cat just reads its standard input and writes it to standard output.
- `head [-n N]` reads its standard input and writes only the first N lines (default is 10 if you leave the option off) to standard output. You can also put a minus before the argument e.g. `head -n -2` to *skip the last 2 lines* and write all the rest.
- `tail [-n N]` is like head except that it writes the last N lines (with a minus, it skips the first N ones).
- `sort` reads all its standard input into a memory buffer, then sorts the lines and writes them all to standard output.
- `uniq` reads standard input and writes to standard output, but skips repeated lines that immediately follow each other, for example if there are three lines A, A, B then it would only write A, B but if it gets A, B, A it would write all three. A common way to remove duplicate lines is `... | sort | uniq | ...`.
- `grep [-iv] EXPRESSION` reads standard input and prints only lines that match the regular expression to standard output. With `i` it is case-insensitive, and with `v` it only prints lines that do *not* match the expression.
- `sed -e COMMAND` reads lines from standard input, transforms them according to the command and writes the results to standard output. `sed` has its own command language but the most common one is `s/SOURCE/DEST/` which changes substrings matching the source regular expression into the destination one.
- `wc [-l]` stands for word count, but with `l` it counts lines instead. Putting a `wc -l` on the very end of a pipe is useful if you just want to know how many results a particular command or pipe produces, assuming the results come one per line.

All these commands actually take an optional extra filename as argument, in which case they read from this file as input. For example, to display the first 10 lines of a file called `Readme.txt`, you could do either `cat Readme.txt | head` or `head Readme.txt`.

Word list exercises - pipes and regular expressions

Most Linux distributions (including Debian) come with a dictionary file `/usr/share/dict/words` that contains a list of English words in alphabetical order, for use in spell-checking programs. The list includes a selection of proper nouns, for example countries and cities. If you want to look at it on a system that doesn't have it, you can download it with:

```
wget https://users.cs.duke.edu/~ola/ap/linuxwords -O words
```

`wget` is one of two utilities for downloading files, the other being `curl`. Note that the option for output file name is a capital O, not a lowercase o or a zero.

Find one-line commands, possibly with pipes, to print the following to your terminal. You can either start each command with `cat /usr/share/dict/words | ...` or do it without cat by providing the words file as an argument to the first command in your pipeline.

- The first word in the file.

```
cat /usr/share/dict/words | grep -i '^[a]' | head -n 1


OUTPUT:

A
```

- The last word in the file.

```
cat /usr/share/dict/words | grep -i '^[z]' | tail -n 1
```

```
OUTPUT:

ZZZ
```

- The number of words in the words file - there is one word per line.

```
cat /usr/share/dict/words | grep -i '^[a-z]' | wc -l


OUTPUT:

479199
```

- The 6171st word in the file.

```
cat /usr/share/dict/words | grep -i '^[a-z]' | head -n 6171 | ta


OUTPUT:

AFB
```

- All words containing the letter Q, capitalised. (A regular expression containing a string of one or more letters matches all strings that contain the expression as a substring.)

```
cat /usr/share/dict/words | grep 'Q'


OUTPUT:
```

```
Output too long to paste - assume it was correct...
```

- All words starting with the letter X. The regular expression `X` would match an X anywhere in the word, but `^X` matches an X only at the start of the string.

```
cat /usr/share/dict/words | grep '^[X]'


OUTPUT:

Output too long to paste - assume it was correct...
```

- All words ending in j. (The expression `'j$'` matches a j only at the end of the string, but you have to single-quote it to stop the shell from interpreting the dollar sign).

```
cat /usr/share/dict/words | grep 'j$'
```

- The number of words containing the letter Q, ignoring case (e.g. capitalised or not).

```
cat /usr/share/dict/words | grep -i '[q]'
```

- The first five words containing the letter sequence `cl`.

```
cat /usr/share/dict/words | grep 'cl' | head -n 5


OUTPUT:

abacli
acanthocladous
acclaim
```

```
acclaimable
acclaimed
```

- All words containing the sequence "kp", but not "ckp".

```
cat /usr/share/dict/words | grep -v 'ckp' | grep 'kp'
```

- The last 15 words of exactly two letters. The expression `.` (period) matches a single character, and `'^..$'` for example would match all strings of the format *exactly three characters between start and end of string*. You need to quote it because of the dollar sign.

```
cat /usr/share/dict/words | grep -i '^..$' | tail -n 15


OUTPUT:



ZB
zB
ZD
ZG
ZI
ZK
Zl
Zn
zn
zo
Zr
zs
ZT
```

```
ZZ
Zz
```

- All words from the first 100 words on the list, which contain the letter y.

```
cat /usr/share/dict/words | head -n 100 | grep 'y'


OUTPUT:

Aaqbiye
```

- The first five words that are among the last 100 words on the list, and contain the letter y (whether capitalised or not).

```
cat /usr/share/dict/words | tail -n 100 | grep -i 'y'| head -n 5


OUTPUT:

Zygosaccharomyces
zygose
zygoses
zygosis
zygosities
```

- All three-letter words with no vowels (aeiou).The regular expression `'[aeiou]'` matches any string that contains one of the bracketed characters; you need quotes to stop the shell from interpreting the brackets. Remember to exclude words with capitalised vowels as well. *There are 343 of these.*

```
cat /usr/share/dict/words | grep -iv '[aeiou]' | grep '^...$' |
```

```
OUTPUT:

1768 NOT 343????
```

- All words of exactly 7 letters, where the third one is an e and the word ends "-ded". *This kind of search is really useful for crosswords. There are 14 words of this form, can you guess them?*

```
cat /usr/share/dict/words | grep -i '^..e....$' | grep 'ded$'


OR

cat /usr/share/dict/words | grep -i '^..e.ded$'

OUTPUT:

amended
bielded
blended
breaded
brended
cleaded
creeded
dreaded
emended
fielded
kneaded
pleaded
scended
shedded
sledded
sleided
snedded
speeded
```

```
steaded
treaded
trended
tweeded
unended
upended
wielded
yielded
```

Bonus regular expression question:

- Find all words that start with a P (whether capitalised or not), and contain at least four instances of the letter a. Putting a `*` after something in a regular expression searches for *any number of repetitions of this, including 0* so for example `'a*'` would find words with any number of the letter a, including 0 (which is not what you want here). You need single quotes to stop the shell from expanding the `*`. *Can you guess the words? There are 14 hits in the solution but essentially five words: two demonyms and three nouns which are not proper nouns, all with possessive and plural forms (bar one which is its own plural).*

```
grep -i '^p.*a.*a.*a.*a' /usr/share/dict/words


?????
```

NEED NOTES ON HOW THIS WORKS!!!