

Lecture 02 - Pipes and filters in Unix

pipes - linking output to input of next function or filter

- ▶ functions transform a stream of input into a stream of output and are referred to as filters
- ▶ pipes stitch these filters together sequentially and facilitate **orthogonal design**
- ▶ when creating your own functions/scripts/programs in Unix it is useful to maintain orthogonal design
 - ▶ take input from `stdin`
 - ▶ write to `stdout`
 - ▶ in other words make filters that can be connected with pipes

assignment or redirection

- ▶ Often we want to capture or store the results of a series of filters and pipes
- ▶ Is `stdout` useful for this purpose?
- ▶ What options do we have for capturing filter-pipe output?
 - ▶ `stdout` vs. redirection to a file
 - ▶ `>`, `<`, `>>`
 - ▶ don't use the same file name as input!!!

pipes - quiz question 2

```
cat numbers.txt | wc > counts.txt
```

pipes - quiz question 2

```
cat numbers.txt | wc > counts.txt
```



pipes - quiz question 2

```
cat numbers.txt | wc > counts.txt
```

cat writes the content of `numbers.txt` to **stdout**, which is **pip**ed as **stdin** to `wc`. The **filter** `wc` counts the number of lines, words, and characters received from **stdin** and writes these values to **stdout**. **stdout** is then redirected to the file `counts.txt`.

building pipelines

- ▶ problem decomposition is key (a theme that will be revisited over and over)
- ▶ step-by-step
- ▶ build sequentially and look at intermediate products at each step

printing and viewing options

- ▶ `cat`
- ▶ `less`
- ▶ `wc`

building pipelines

```
cat animals.txt
```

```
cat animals.txt | wc
```

```
cat animals.txt | cut -d , -f 2 | less
```

```
cat animals.txt | cut -d , -f 2 | wc
```

```
cat animals.txt | cut -d , -f 2 | sort -r | less
```

```
cat animals.txt | cut -d , -f 2 | sort -r > final.txt
```

```
cat final.txt
```

build a pipeline

Given the file `animals.txt`. What are the number of unique animal observations across the data collection?

automate the process for files of the same form

Wildcards (*, ?) provide an opportunity to evaluate a number of files with similar structure all at once.

This is powerful, but with great power comes great responsibility. . .

function review

wc

sort

head

tail

uniq

cut