Creating small tools in C

Lecture 02.02

Working towards Assignment 1

- Small tools written in C perform specialized small tasks, such as reading and writing files, or filtering data
- If you want to perform more complex tasks, you can link several tools together

```
head A | sort > first10A.output
```

But how are these small tools built?

We will find out together

Demo: my travel app

- We developed a map application to display location-aware travel stories
- We collect the data using GPS device
- We download the data from the device, add info about each location, and it is now in the following format:

```
42.363400,-71.098465,Ate dinner
42.363327,-71.097588,Met a cute dog
42.363255,-71.096710,Took a streetcar
```

The **input** is in a comma-delimited (csv) format

To display data on the map:

The **input** is in a comma-delimited (csv) format

```
42.363400,-71.098465,Ate dinner
42.363327,-71.097588,Met a cute dog
42.363255,-71.096710,Took a streetcar
```

We need to convert the input into a format compatible with map API:

```
var data=[
    {latitude: 42.363400, longitude: -71.098465, info: 'Ate dinner'},
    {latitude: 42.363327, longitude: -71.097588, info: 'Met a cute dog'},
    {latitude: 42.363255, longitude: -71.096710, info: 'Took a streetcar'},
    ...
];
```

JavaScript Object Notation (JSON) format

Small tool: data format converter

- Tools that read data line by line, process it, and write something out are called ***filters*** (head, tail, unix2dos, sed)
- Our problem is a good candidate for developing a new filter: **geo2js**
- It will take as an input the lines with a predefined format, extract data pieces into variables, and print the variable values in a new format

scanf and *printf*

Working on starter code in geo2js.c

```
gcc geo2js.c -o geo2js && ./geo2js
```

- When we compile and run we can input lines from the command line:

```
40.5,-70.5,Place 1
```

- Press ctrl+D to terminate

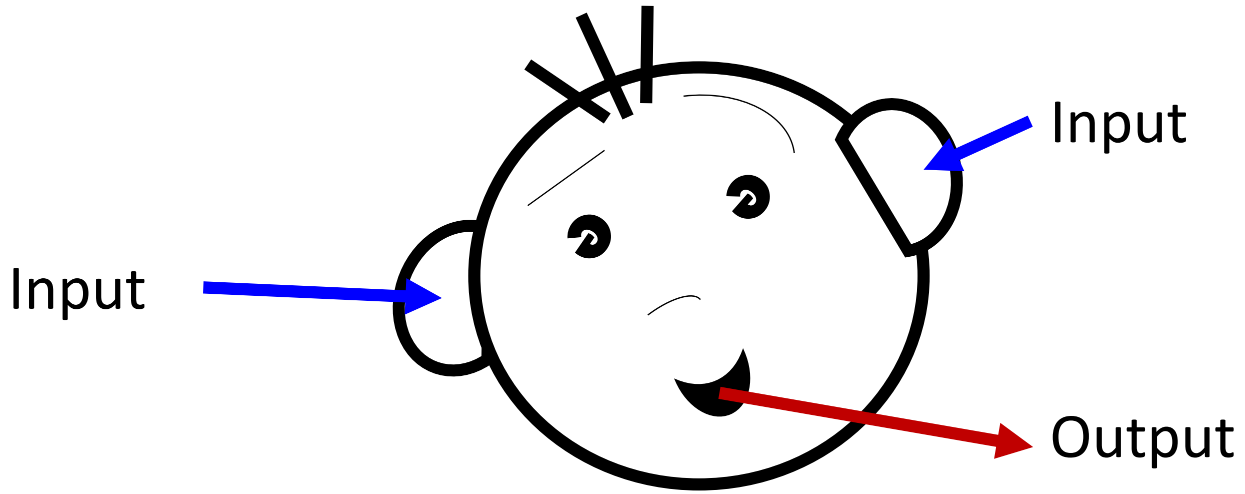
We need to read and write from/to file

- We want to get large amounts of data by reading a file
- We want to output JSON array to a file called output.js

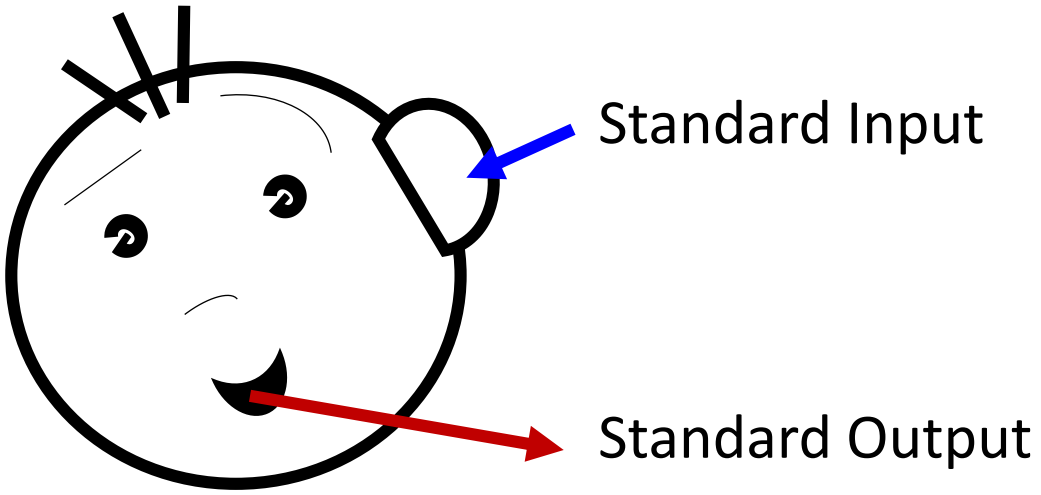
- What code do we have to change?
- Do we have to change any code at all?

Magic of *scanf* and *printf*

- The truth: *scanf* and *printf* do not really talk to the keyboard and the display
- They talk to the Standard Input and Standard Output
- The Standard Input and Standard Output are created by the operating system when the program runs and is set by default to the keyboard and the display



Human



Process

Redirecting standard input and standard output

- The operating system controls how data gets into and out of the Standard Input and Output
- The scanf() and printf() don’t know, or care, where the data comes from or goes to. They just read and write Standard Input and the Standard Output
- We can *redirect* the Standard Input and Standard Output so that they read and write data somewhere else, such as to and from files

```
geo2js < walk.csv
geo2js < walk.csv > output.js
```

We created a valid output.js file

Test: mywalk.html in a modern browser

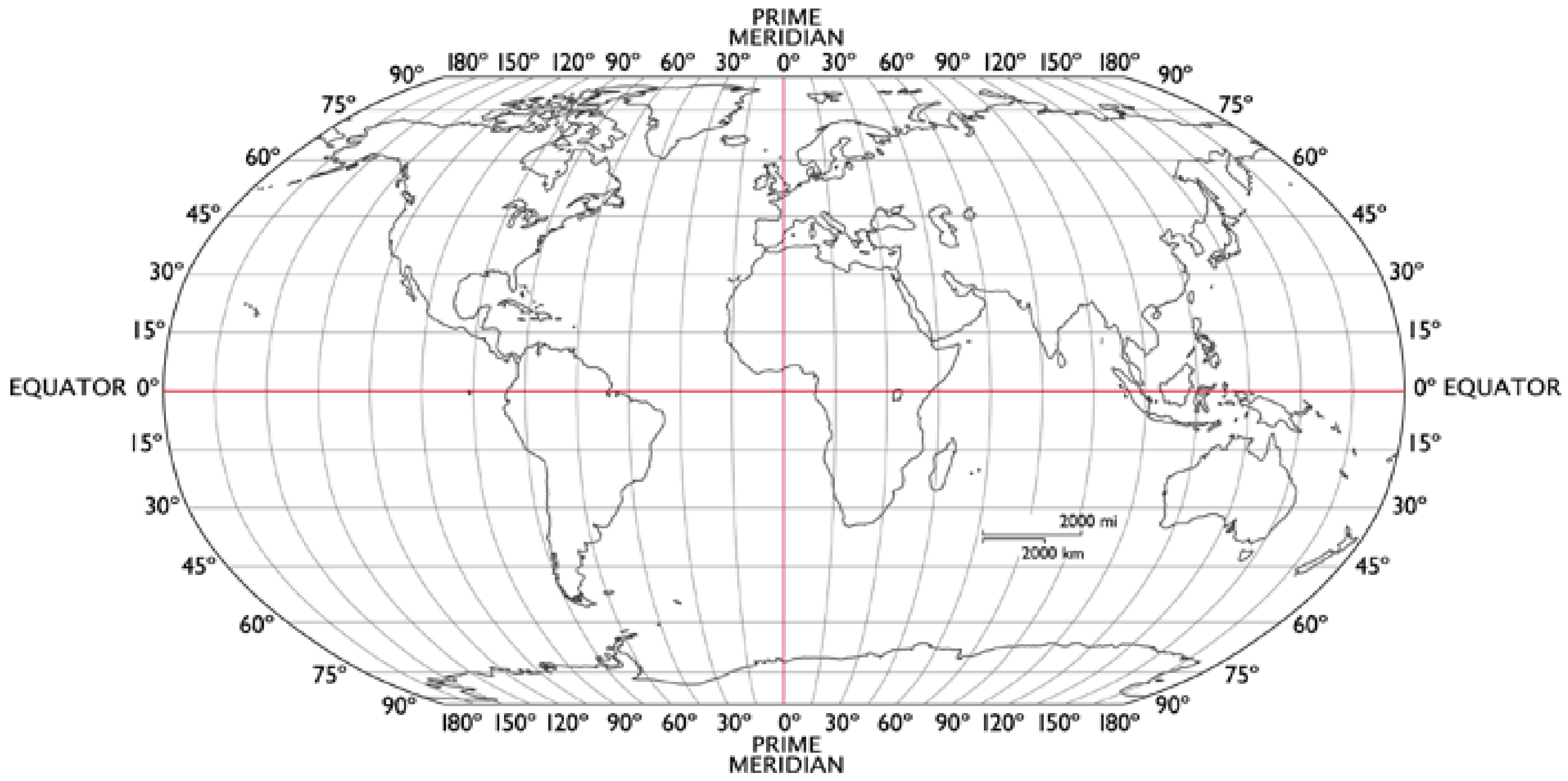
Invalid input

- I have dropped my GPS unit several times and now the data won't display

- How does the new data look like:

```
430.664514,-79.380448,Pub OGrady brunch
43.664351,-79.380362,Nighty DJ at Woodys
43.664153,-790.380265,Kintaro Izakaya eat some sushi
```

INVALID GPS COORDINATES!



Valid range from 0° to (+/–)90°

Latitude

Valid range from 0° to (+/–)180°

Longitude

Implementing data validation

- That should be easy to fix: If a latitude or longitude falls outside the expected numeric, just display an error message and skip this line

Fix the code to perform data validation

Run the code

- Why did map application stop working?
- Why does it think that the entire output.js file was corrupt?

- Why weren’t there any error messages?

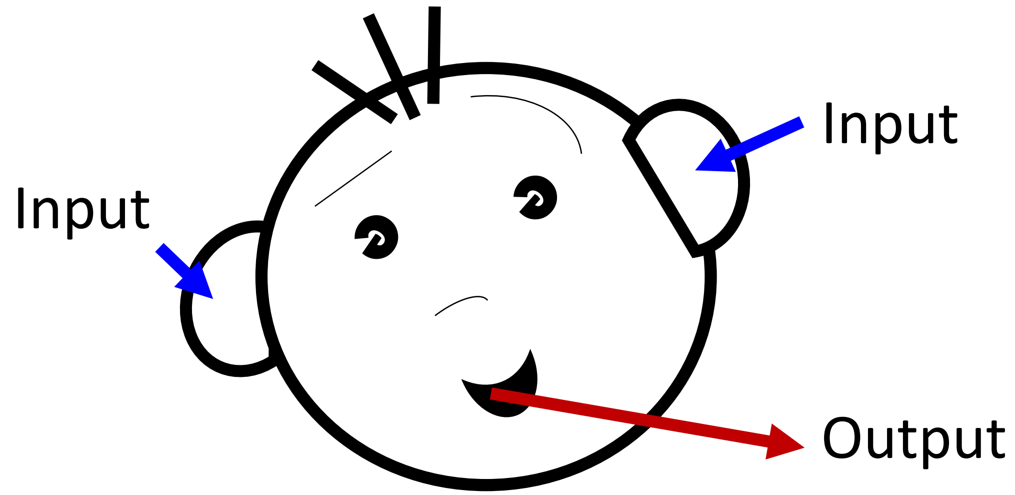
Hint: look inside output.js for clues

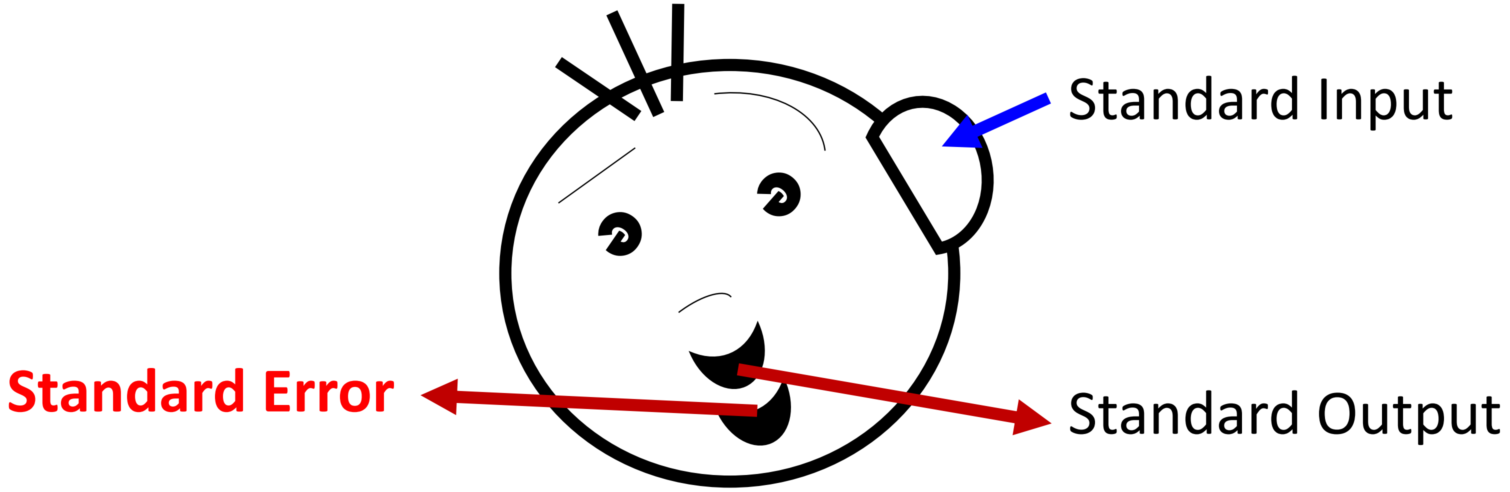
The problem with redirection

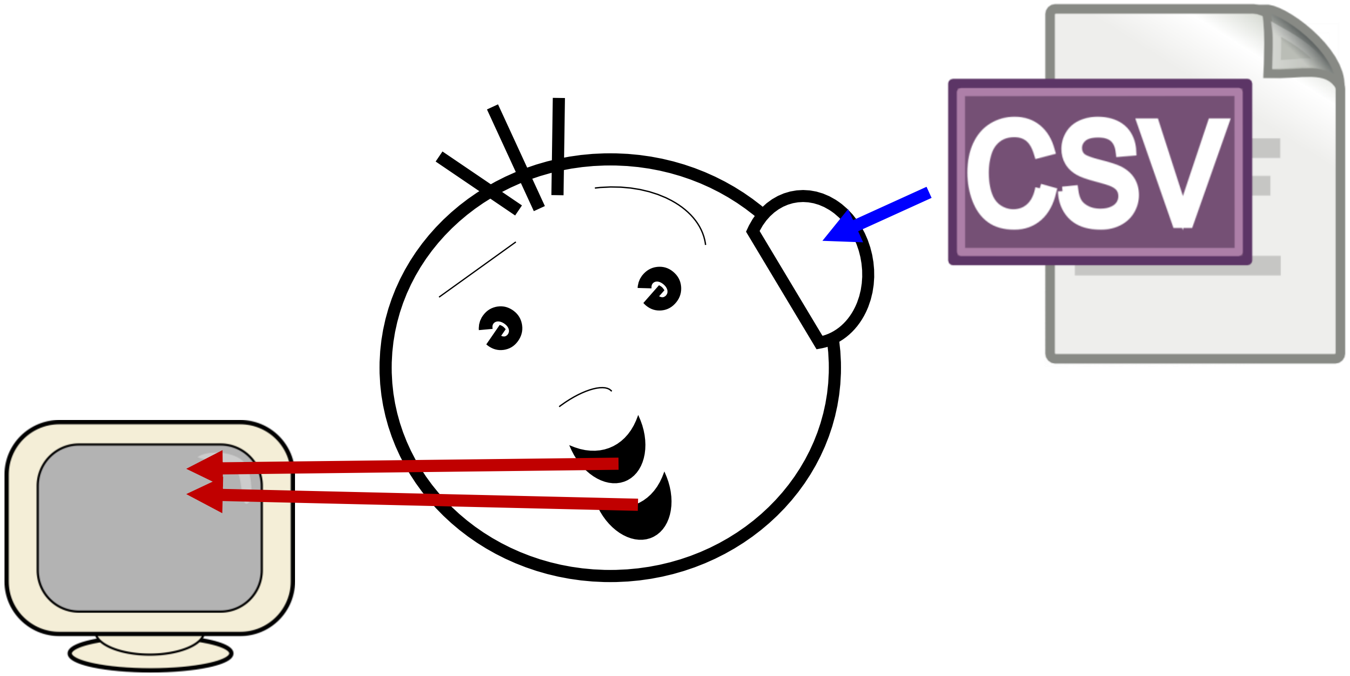
- Because we were redirecting the Standard Output into the output.js, we were also redirecting the error messages to the same file

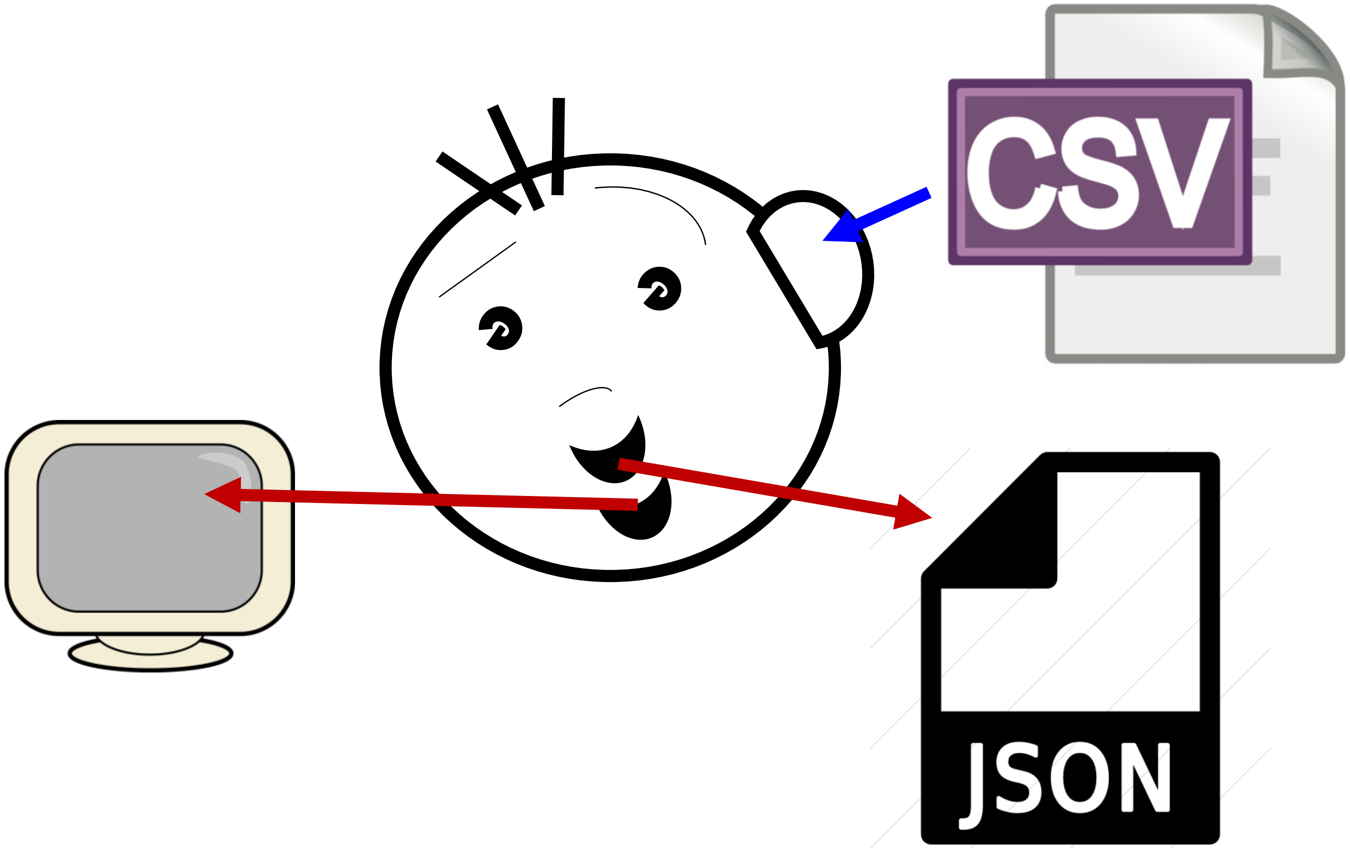
- Can we still display error messages on the screen if we are redirecting the output to the file?
- Is there a special output for errors so we do not mix errors with Standard Output?

Human vs. Process






CSV


CSV
JSON

fprintf() prints to a data stream

```
printf ("I like Turtles!");
```

- This is EXACTLY the same as:

```
fprintf (stdout, "I like Turtles!");
```

- This prints to a different stream:

```
fprintf (stderr, "Invalid data format!");
```

- There is also stdin:

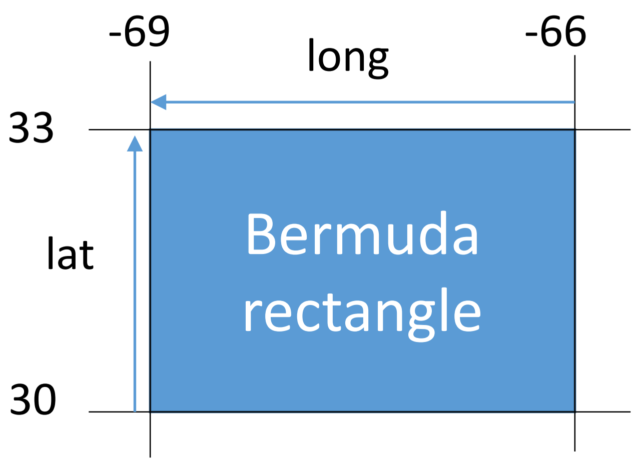
```
fscanf(stdin, ...)
```

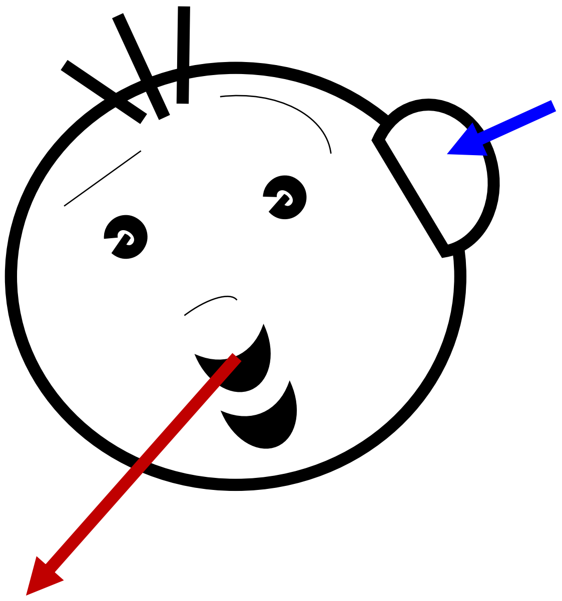
> redirects stdout
2> redirects stderr

```
geo2js 2> errors.txt
```

Update the code with fprintf and fscanf

Writing second filter



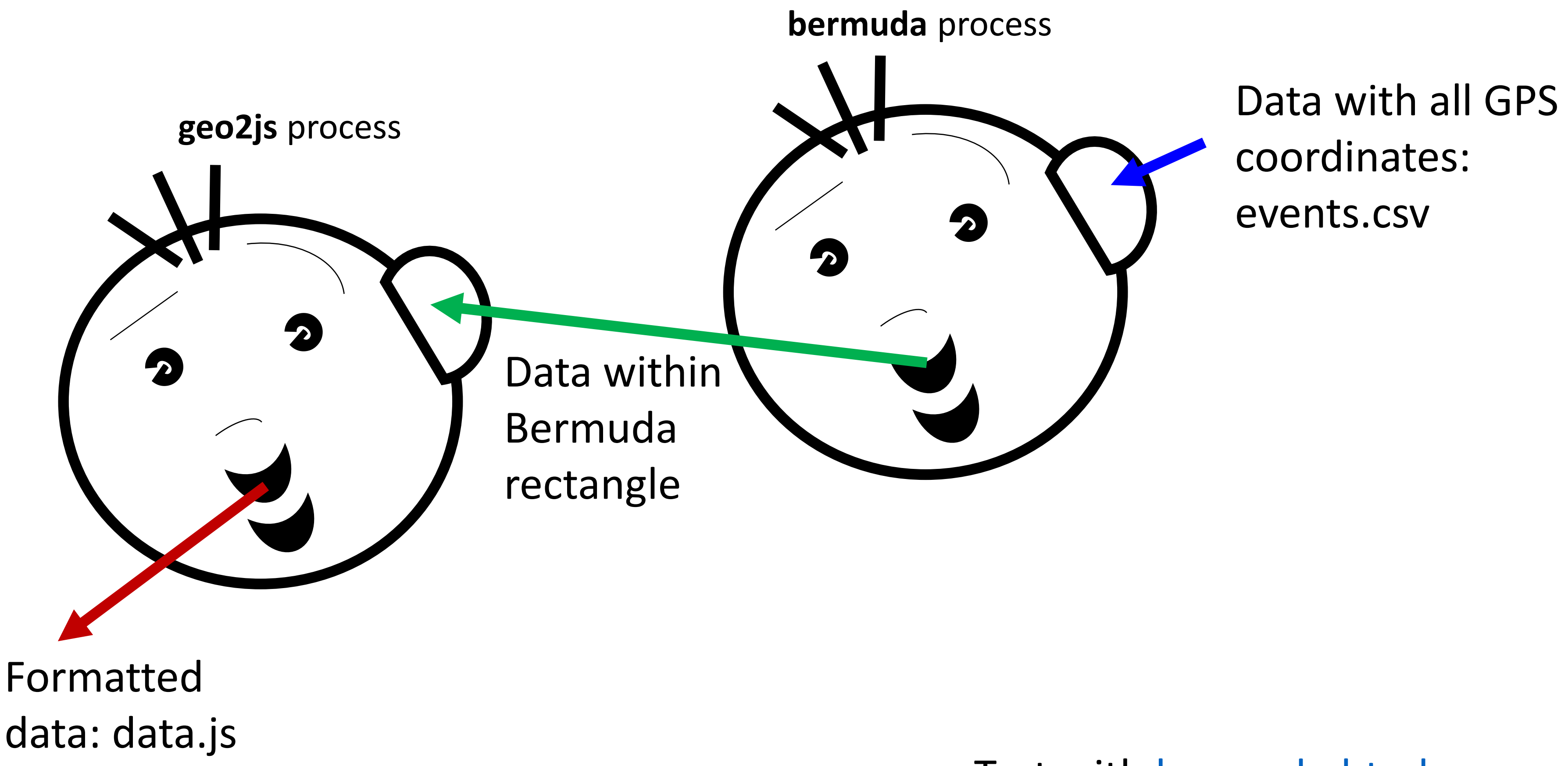


Data with all GPS coordinates

Data within Bermuda rectangle

Connecting 2 processes with a pipe

```
(./bermuda < events.csv) | geo2js > data.js
```


bermuda process
Data with all GPS coordinates: events.csv
geo2js process
Data within Bermuda rectangle
Formatted data: data.js

Test with [bermuda.html]()

Pipes

- Each pipe accepts data at one end, and sends the data out of the other end *in sequence*
- Both of the programs run *at the same time*: as output is produced by the first process, it can be consumed by the second process
- We can connect more than 2 programs together with pipes. A series of connected processes is called a *pipeline*

Small tools: summary

- Small tools usually solve a small technical problem, like converting data from one format to another. If you can combine them together, then you can solve large problems
- Small tools should use standard input and standard output to make it easy to connect them together and redirect input/output to a file
- Small tools work with text files: It's the most open format, and other programmer can easily read and understand the output
- If you want to perform a different task, consider writing a separate small tool and connect it with existing tools using pipes