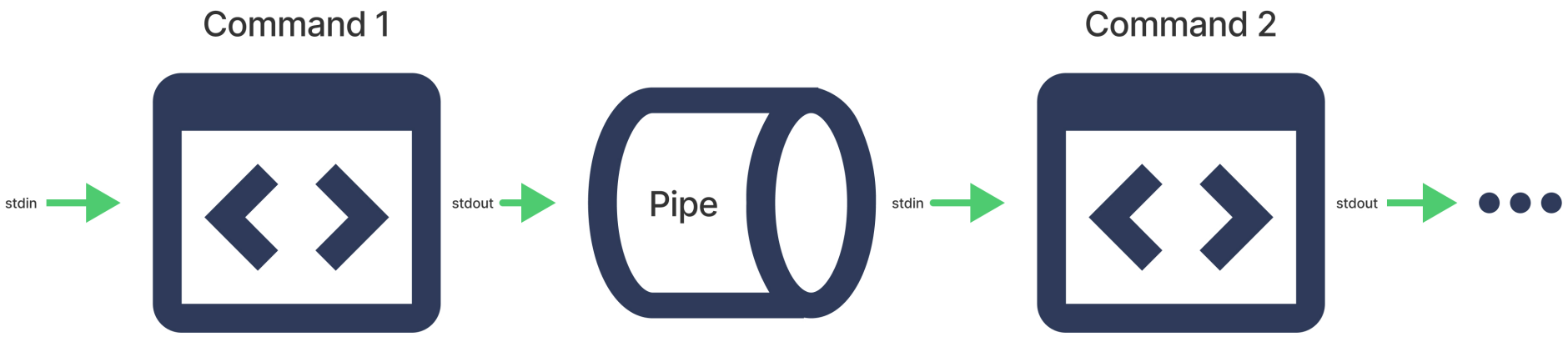
Pipes



What are pipes?

Put simply, pipes are commands in Linux which allow you to use the output of one command as the input of another.

Pipes "`|`" use the following format:

```
[command 1] | [command 2] | [command 3] ... | [command n]
```

(no limit to the number of times in a row u can pipe!)

Pipe examples

Ex 1:

Let's start with a commonly used example.

Recall the commands:

- `sort`: sorts lines in input
- `uniq`: prints input with consecutive repeated lines collapsed to a single, unique line

With the help of the pipe operator, you can combine these commands to print all the unique lines in a file!

Suppose you have the file `pets.txt` with the following contents:

```
$ cat pets.txt
goldfish
dog
cat
parrot
dog
goldfish
goldfish
```

If you *only* use `sort` on pets.txt, you get:

```
$ sort pets.txt
cat
dog
dog
goldfish
goldfish
goldfish
parrot
```

And if you *only* use `uniq`, you get:

```
$ uniq pets.txt
goldfish
dog
cat
parrot
dog
goldfish
```

But by combining the two commands in the correct order, you get back:

```
$ sort pets.txt | uniq
cat
dog
goldfish
parrot
```

which are the sorted, unique lines from `pets.txt`!

Ex 2:

Some commands, such as `tr`, *only* accept "standard input" as input (not strings or filenames):

- `tr` (translate) - replaces characters in input text.
  - Syntax: `tr [OPTIONS] [target characters] [replacement characters]`

In cases like this, we can use piping to apply the command to strings and file contents.

With strings, you could, for example, use `echo` in combination with `tr` to replace all vowels in a string with underscores, as follows:

```
$ echo "Linux and shell scripting are awesome\!" | tr "aeiou" "_"
L_n_x _nd sh_ll scr_pt_ng _r_ _w_s_m_!
```

To perform the complement of the operation from the previous example, that is, to replace all consonants with an underscore, you can use the -c option like this:

```
$ echo "Linux and shell scripting are awesome\!" | tr -c "aeiou" "_"
_i_u__a_____e_____i__i___a_e_a_e_o_e_
```

With files, you could use `cat` in combination with `tr` to change all of the text to upper case as follows:

```
$ cat pets.txt | tr "[a-z]" "[A-Z]"
GOLDFISH
DOG
CAT
PARROT
DOG
GOLDFISH
GOLDFISH
```

The possibilities are endless! For example:

```
$ sort pets.txt | uniq | tr "[a-z]" "[A-Z]"
CAT
DOG
GOLDFISH
PARROT
```

Ex 3:

You can even use curl in combination with the grep command to extract components of URL data by piping the output of curl to grep.
Let's see how you can use this pattern to get the current price of BTC (Bitcoin) in USD.

First, you find a public URL API. In this example, you will use one provided by CoinStats.

Specifically, they provide a public API (no key required) `https://api.coinstats.app/public/v1/coins/bitcoin\?currency\=USD` which returns some json about the current BTC price in USD.

You can see what this looks like in your browser:


```
{"coin":
{"id":"bitcoin","icon":"https://static.coinstats.app/coins/Bitcoin6l39t.png","name":"Bitcoin","symbol":"BTC","rank":1,"price":58241.582859709466,"price
Btc":1,"volume":42923267434.05484,"marketCap":1099617799725.5955,"availableSupply":18880287,"totalSupply":21000000,"priceChange1h":0.92,"priceChange1d"
:-0.97,"priceChange1w":-11.06,"websiteUrl":"http://www.bitcoin.org","twitterUrl":"https://twitter.com/bitcoin","exp":
["https://blockchair.com/bitcoin/","https://btc.com/","https://btc.tokenview.com/"]}}
```


Entering the following command returns the BTC price data, displayed as a json object:


```
$ curl -s --location --request GET https://api.coinstats.app/public/v1/coins/bitcoin\?currency\=USD
{
  "coin": {
    "id": "bitcoin",
    "icon": "https://static.coinstats.app/coins/Bitcoin6l39t.png",
    "name": "Bitcoin",
    "symbol": "BTC",
    "rank": 1,
    "price": 57907.78008618953,
    "priceBtc": 1,
    "volume": 48430621052.9856,
    "marketCap": 1093175428640.1146,
    "availableSupply": 18877868,
    "totalSupply": 21000000,
    "priceChange1h": -0.19,
    "priceChange1d": -0.4,
    "priceChange1w": -9.36,
    "websiteUrl": "http://www.bitcoin.org",
    "twitterUrl": "https://twitter.com/bitcoin",
    "exp": [
      "https://blockchair.com/bitcoin/",
      "https://btc.com/",
      "https://btc.tokenview.com/"
    ]
  }
}
```


(The output is formatted for your convenience in this lab).

The json field you want to grab here is `"price": [numbers].[numbers]"`. To grab this you can use the following grep command to extract it from the json text:

```
grep -oE "\"price\"\s*:\s*[0-9]*?\.[0-9]*"
```

Let's break down the details of this statement:

- `-o` tells `grep` to *only* return the matching portion
- `-E` tells `grep` to be able to use extended regex symbols such as `?`
- `\"price\"` matches the string `"price"`
- `\s*` matches any number (including 0) of whitespace (`\s`) characters
- `:` matches `:`
- `[0-9]*` matches any number of digits (from 0 to 9)
- `?\.` optionally matches a `.` (this is in case price were an integer)

Now that you have the grep statement that you need, you can pipe the BTC data to it using the curl command from above:

```
$ curl -s --location --request GET https://api.coinstats.app/public/v1/coins/bitcoin\?currency\=USD |\
    grep -oE "\"price\":\s*[0-9]*?\.[0-9]*"
"price": 57907.78008618953
```

The backslash `\` character used here after the pipe `|` allows you to write the expression on multiple lines.

Finally, to get *only* the value in the price field, and drop the "price" label, you can use chaining to pipe the same output to another `grep`:

```
$ curl -s --location --request GET https://api.coinstats.app/public/v1/coins/bitcoin\?currency\=USD |\
    grep -oE "\"price\":\s*[0-9]*?\.[0-9]*" |\
    grep -oE "[0-9]*?\.[0-9]*"
57907.78008618953
```

Beautiful 😎