Skills Network

# Examples of Pipes

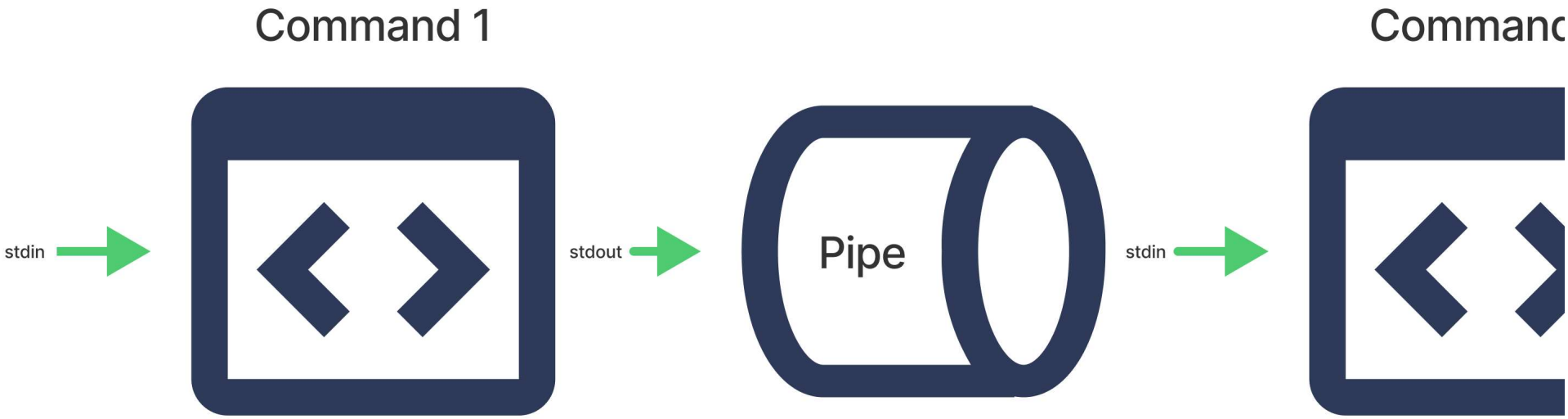
## Learning Objectives

After completing this reading, you will be able to:

- Describe pipes
- Use pipes to combine commands when working with strings and text file contents
- Use pipes to extract information from URLs

## 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:

```
1. 1
```

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

Copied!

There is no limit to the number of times you can chain pipes in a row!

In this lab, you'll take a closer look at how you can use pipes and filters to solve basic data processing problems.

## Pipe examples

### Combining commands

Let's start with a commonly used example. Recall the following commands:

- `sort` - sorts the lines of text in a file and displays the result
- `uniq` - prints a text file with any consecutive, repeated lines collapsed to a single 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:

```
1
2
3
4
5
6
7
8
```

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

Copied!

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

```
1
2
3
4
5
6
7
8
```

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

Copied!

The file is sorted, but there are duplicated lines of "dog" and "goldfish".

On the other hand, if you *only* use `uniq`, you get:

```
1
2
3
4
5
6
7
```

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

Copied!

This time, you removed consecutive duplicates, but non-consecutive duplicates of "dog" and "goldfish" remain.

But by combining the two commands in the correct order - by first using `sort` then `uniq` - you get back:

```
1
2
3
4
5
```

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

Copied!

Since `sort` sorts all identical items consecutively, and `uniq` removes all consecutive duplicates, combining the commands prints only the unique lines from `pets.txt`!

## Applying a command to strings and files

Some commands such as `tr` only accept *standard input* - normally text entered from your keyboard - but not strings or filenames.

- `tr` (translate) - replaces characters in input text

```
1
```

```
tr [OPTIONS] [target characters] [replacement characters]
```

Copied!

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

With strings, you can use `echo` in combination with `tr` to replace all the vowels in a string with underscores `_`:

```
1
2
```

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

Copied!

To perform the complement of the operation from the previous example - or to replace all the *consonants* (any letter that is not a vowel) with an underscore - you can use the `-c` option:

```
1
2
```

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

Copied!

With files, you can use `cat` in combination with `tr` to change all of the text in a file to uppercase as follows:

```
1
2
3
4
5
6
7
8
```

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

```
5. PARROT
6. DOG
7. GOLDFISH
8. GOLDFISH
```

Copied!

The possibilities are endless! For example, you could add `uniq` to the above pipeline to only return unique lines in the file, like so:

```
1. 1
2. 2
3. 3
4. 4
5. 5
```

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

Copied!

## Extracting information from URLs

You can also 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 Bitcoin (BTC) in USD.

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

CoinStats provides a public API with no key required at `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 by entering the above link in your browser.

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

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
```

```
1. $ curl -s --location --request GET https://api.coinstats.app/public/v1/coins/bitcoin\?currency\=USD
2. {
3.   "coin": {
4.     "id": "bitcoin",
5.     "icon": "https://static.coinstats.app/coins/Bitcoin6139t.png",
6.     "name": "Bitcoin",
7.     "symbol": "BTC",
8.     "rank": 1,
9.     "price": 57907.78008618953,
10.     "priceBtc": 1,
11.     "volume": 48430621052.9856,
12.     "marketCap": 1093175428640.1146,
13.     "availableSupply": 18877868,
14.     "totalSupply": 21000000,
15.     "priceChange1h": -0.19,
16.     "priceChange1d": -0.4,
17.     "priceChange1w": -9.36,
18.     "websiteUrl": "http://www.bitcoin.org",
19.     "twitterUrl": "https://twitter.com/bitcoin",
20.     "exp": [
21.       "https://blockchair.com/bitcoin/",
22.       "https://btc.com/",
23.       "https://btc.tokenview.com/"
24.     ]
25.   }
26. }
```

Copied!

> **Note:** For the purpose of this reading, we've reformatted the output to make it easier to interpret. The actual output is a continuous stream of text.

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

```
1. 1
```

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

Copied!

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 `.`

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

```
1. 1
2. 2
3. 3
```

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

Copied!

> **Tip:** 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`:

```
1. 1
2. 2
3. 3
4. 4
```

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

```
3.     grep -oE "[0-9]*?\.[0-9]*"
4. 57907.78008618953
```

Copied!

This now displays only the numerical price without the label.

## Summary

In this reading, you learned that:

- Pipes are commands in Linux which allow you to use the output of one command as the input of another
- You can combine commands such as `sort` and `uniq` to organize strings and text file contents
- You can pipe the output of a `curl` command to `grep` to extract components of URL data

## Authors

Jeff Grossman
Sam Prokopchuk

## Change Log

| Date (YYYY-MM-DD) | Version | Changed By | Change Description |
|---|---|---|---|
| 2023-05-23 | 1.4 | Benny Li | QA pass |
| 2023-04-27 | 1.3 | Nick Yi | QA pass |
| 2023-04-14 | 1.2 | Nick Yi | ID Review |
| 2023-03-08 | 1.1 | Jeff Grossman | Added new content |
| 2021-11-09 | 1.0 | Sam Prokopchuk | Initial version created |