

# Examples of Pipes

## Learning Objectives

After completing this reading, you will be able to:

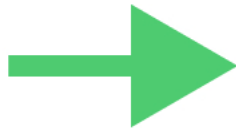
- Describe pipes
- Use pipes to combine commands when working with strings and text file contents
- Use pipes to extract information from URLs

## 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.

# Command

stdin



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

1. $ cat pets.txt
2. goldfish
3. dog
4. cat
5. parrot
6. dog
7. goldfish
8. goldfish
```

Copied!

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

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

1. $ sort pets.txt
2. cat
3. dog
4. dog
5. goldfish
6. goldfish
7. goldfish
8. 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. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7

1. $ uniq pets.txt
2. goldfish
3. dog
4. cat
5. parrot
6. dog
7. 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. 1
2. 2
3. 3
4. 4
5. 5

1. $ sort pets.txt | uniq
2. cat
3. dog
4. goldfish
5. 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. 1
```

```
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. 1
2. 2
```

```
1. $ echo "Linux and shell scripting are awesome\!" | tr "aeiou" "_"
2. 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. 1
2. 2
```

```
1. $ echo "Linux and shell scripting are awesome\!" | tr -c "aeiou" "_"
2. _i_u_a____e____i_i__a_e_a_e_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. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
```

```
1. $ cat pets.txt | tr "[a-z]" "[A-Z]"
2. GOLDFISH
3. DOG
4. 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 JSON Files:

Let's see how you can use this json file to get the current price of Bitcoin (BTC) in USD, by using `grep` command.

```
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
```

```

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

```

Copied!

Copy the above output in a file and name it as Bitcoinprice.txt.

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 .

Use the cat command to get the output of the JSON file and pipe it with the grep command to get the required output.

```

1. 1
1. cat Bitcoinprice.txt | grep -oE "\"price\"\\s*:\\s*[0-9]*?\\. [0-9]*"

```

Copied!

You can also extract information directly from URLs and retrieve any specific data using such grep commands.

▼ Click here to see the process of extracting information directly from URLs and retrieving specific data:

### Extracting information from URLs:

1. Open your web browser and navigate to the following link: <https://openapi.coinstats.app>. You will be redirected to the login page, as shown below:

# Login

Enter Your credential to login into your account.



Log In

Don't have an account? [Create Account](#)

2. Enter your email address and password, then proceed to verify your email. After verification, you will be redirected to the dashboard.
3. On the dashboard, locate and click the Generate API key button. This will generate an API key, as displayed below. Ensure to save this API key.

Dashboard

Usage History

Pricing

API Docs

## Dashboard

2/1,000,000

Your monthly API credit limit

frlKEQvwJxJPAGWfpmpaAlkxoW6GLPU30810IA8



4. Next, click on the API Docs section on the left side of the dashboard.

Dashboard

Usage History

Pricing

API Docs



## Dashboard

2/1,000,000

Your monthly API credit limit

5. By clicking on it, you will be directed to the page shown below:



JUMP TO

CTRL-/

## COINSTATS PUBLIC API

### Market Data

/coins

GET

/coins/{coinId}

GET

/coins/{coinId}/charts

GET

/coins/price/avg

GET

/coins/price/exchange

GET

/tickers/exchanges

GET

/tickers/markets

GET

/fiats

GET

/markets

GET

### Wallet Data

### Exchange Connection

### NFTs

### News

### Transaction

Powered by **readme**

## 2 credits per request

This endpoint allows you to retrieve a list of cryptocu

🔑 LOG IN TO SEE FULL REQUEST HISTORY

TIME	STATUS	USER AGENT
1/8/2024 05:24 PM	● 200	<a href="#">Try It!</a>
1/8/2024 03:35 PM	● 200	<a href="#">Try It!</a>

2 Requests This Month

### QUERY PARAMS

**page** number

**limit** number

**currency** string

**blockchain** string

### RESPONSES

● 200

Get coins list

6. On the right corner of this page, you will find a curl command for retrieving the output in JSON format. To test this functionality, click on the `Try it` button.

- Note: This process works only when you enter the generated API Key into the text box named header requesting Authorization.

7. After Authorization, the output will be generated in JSON format, as below:



## STORY

Make a request to see history.

Month

The screenshot displays a REST client interface with three main sections: AUTHORIZATION, REQUEST, and RESPONSE.

- AUTHORIZATION:** The 'Header' tab is active, showing a single header: `MoN7ze0A1vuNY6/66FrVZaqMZtCicFDw1JWRDuR4tT8=`. This header is highlighted with a red box.
- REQUEST:** The 'CURL' tab is active, showing the following command:

```
1 curl --request GET \  
2 --url https://openapiv1.coinstats.app/coins \  
3 --header 'X-API-KEY: MoN7ze0A1vuNY6/66FrVZaqMZtCicFDw1JWRDuR4tT8=' \  
4 --header 'accept: application/json'
```

A red box highlights the 'Try It!' button in the bottom right corner of the request section.
- RESPONSE:** The 'RESPONSE' tab is active, showing a JSON response. A red arrow points from the 'Try It!' button to this section. The response is as follows:

```
1 {  
2   "result": [  
3     {  
4       "id": "bitcoin",  
5       "icon": "https://static.coinstats.app/coins/16504",  
6       "name": "Bitcoin",  
7       "symbol": "BTC",  
8       "rank": 1,  
9       "price": 46560.58965499212,  
10      "priceBtc": 1,  
    }  
  ]  
}
```

## Summary

- 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
2024-01-09	1.5	Zehra Afzal, K Sundararajan	Content corrected after deprecation of Bitcoin API
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

Copyright (c) 2024 IBM Corporation. All rights reserved.