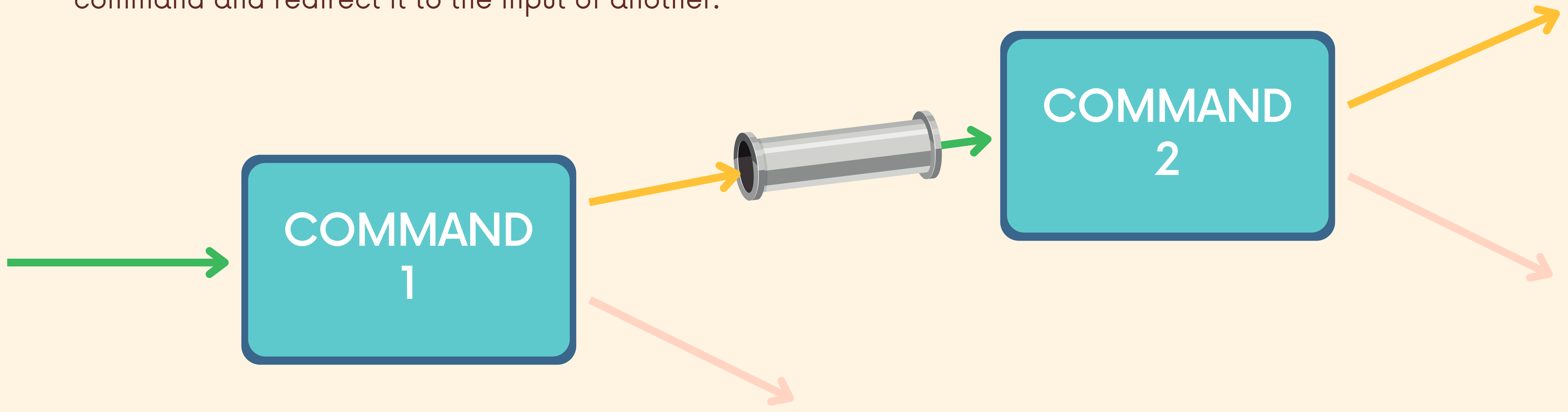
Piping

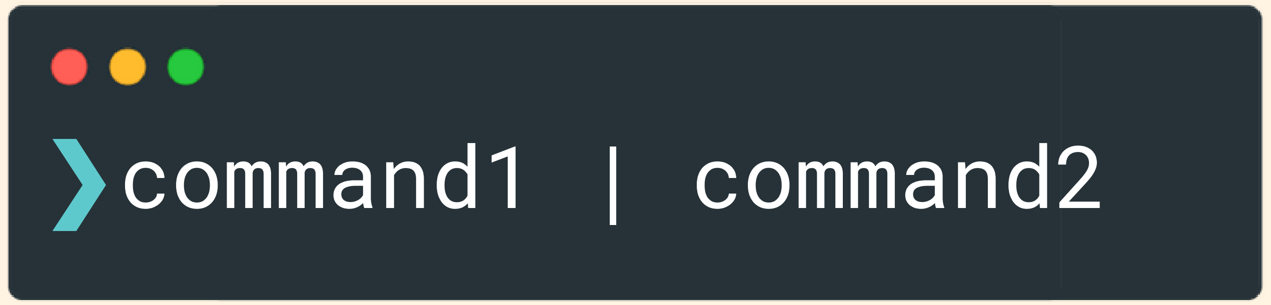
Pipes

Pipes are used to redirect a stream from one program to another program. We can take the output of one command and redirect it to the input of another.

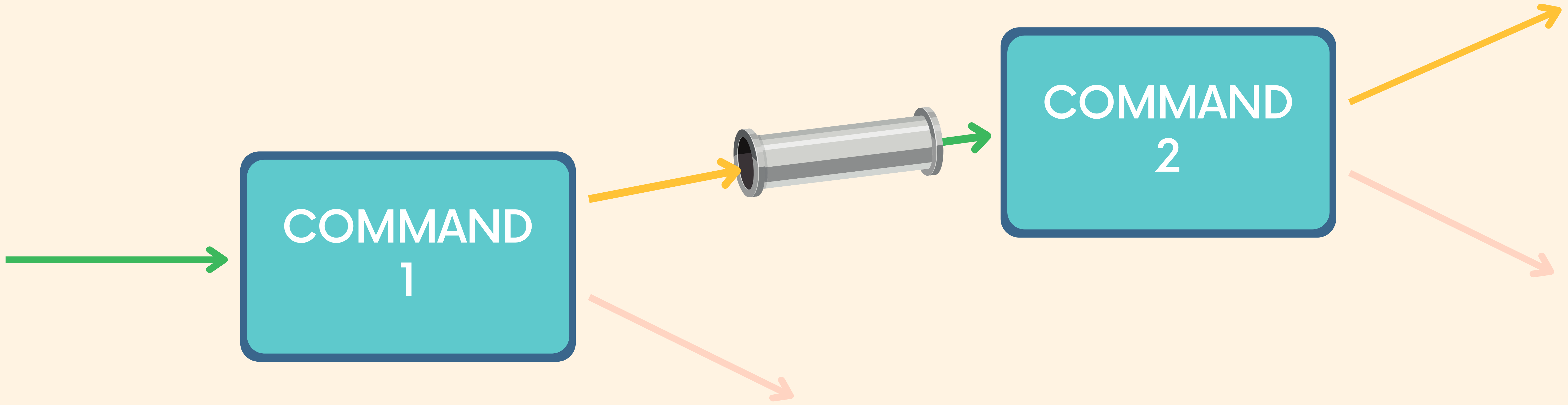


The Syntax

We use the pipe character (|) to separate two commands. The output of the first command will be passed to the standard input of the second command.

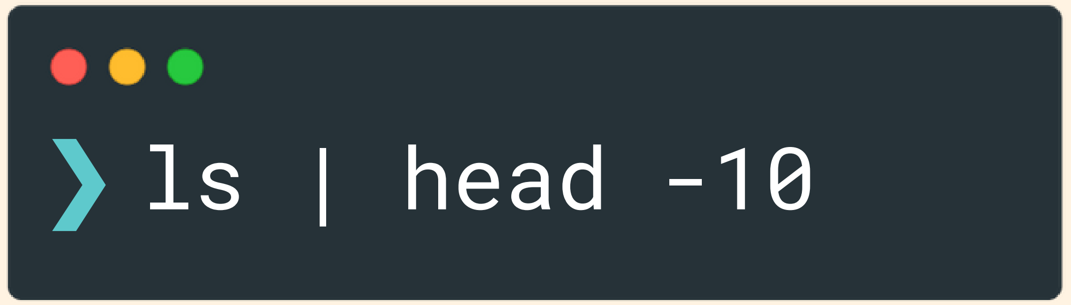


```
❯command1 | command2
```

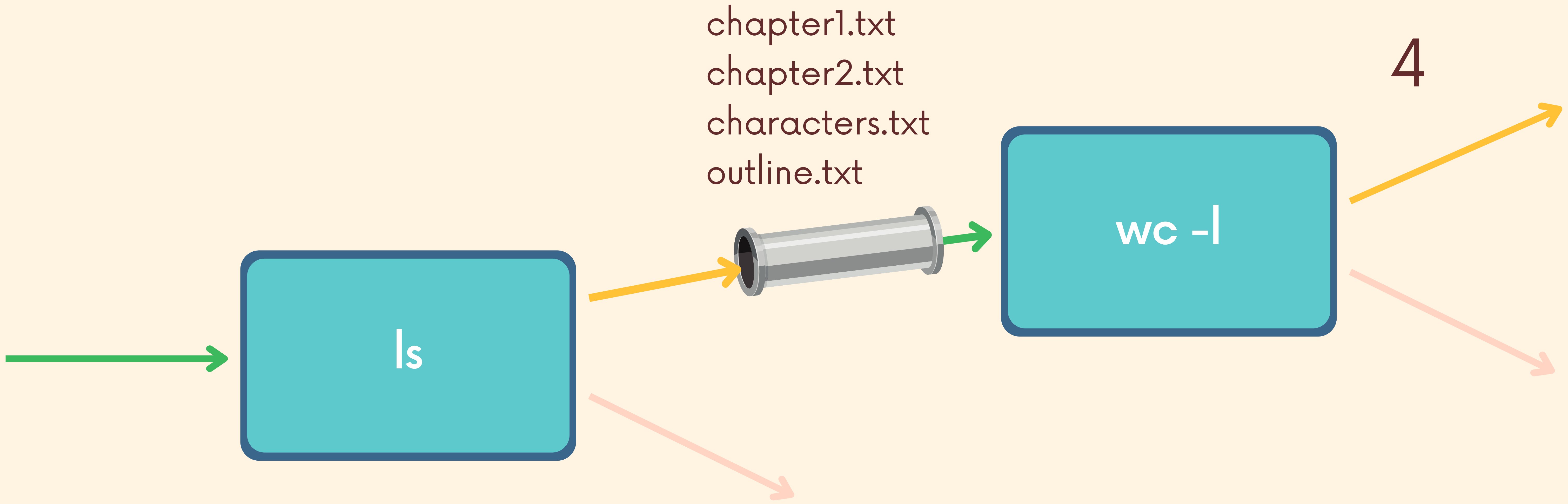


ls | head

This example list the files (non hidden files) in a directory. We pipe the output of ls to the word count command. The -l option tells wc to count the number of lines.

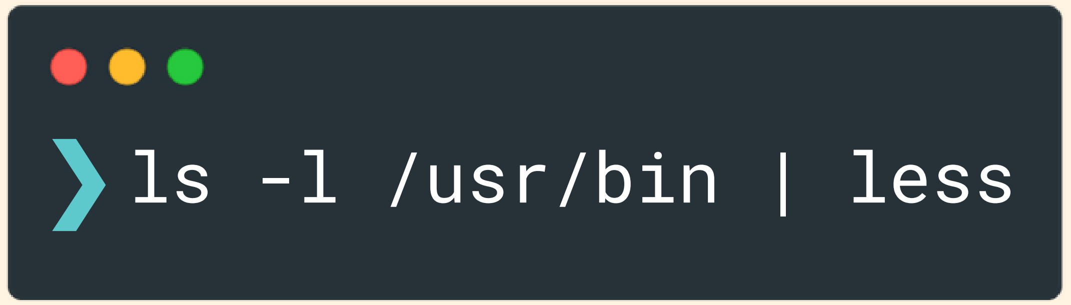


```
❯ ls | head -10
```

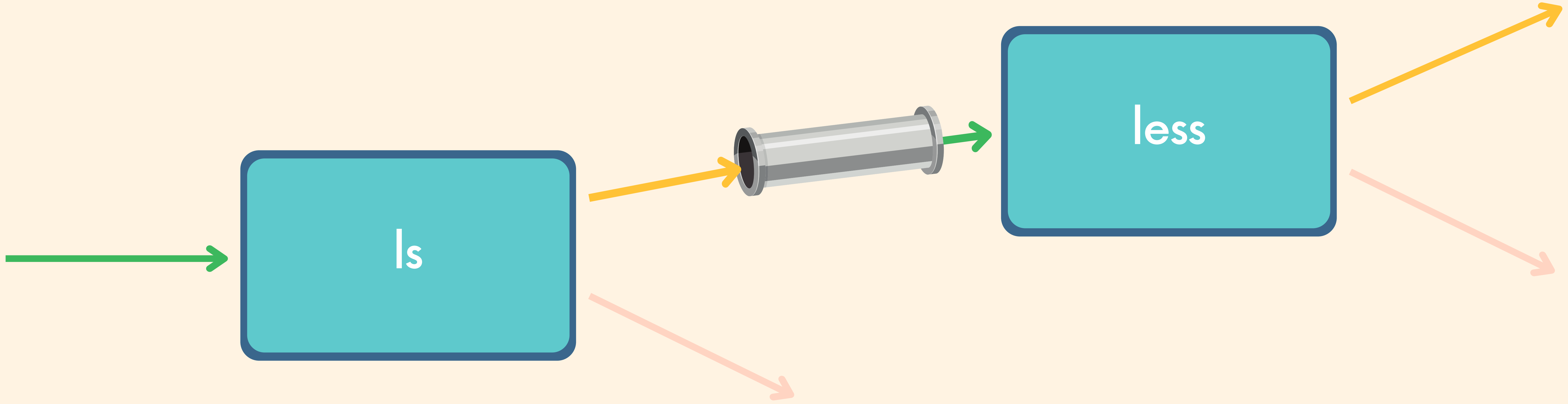


ls | less

This example pipes the output of **ls** to **less**. the /usr/bin directory typically contains a bunch of stuff, so it can be nice to use less to read the results in a more manageable way.



```
❯ ls -l /usr/bin | less
```

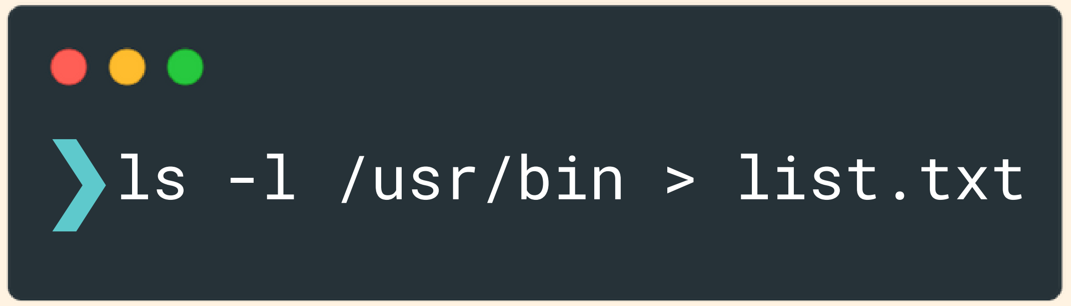


> vs |

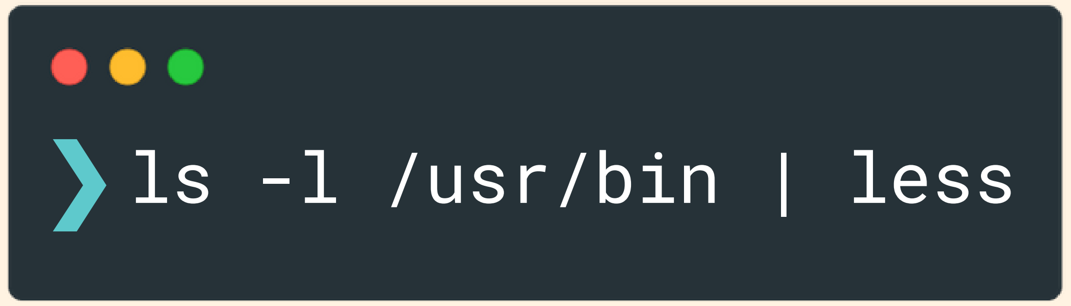
Though both the > character and the | character are used to redirect output, they do it in very different ways.

> connects a command to some file.

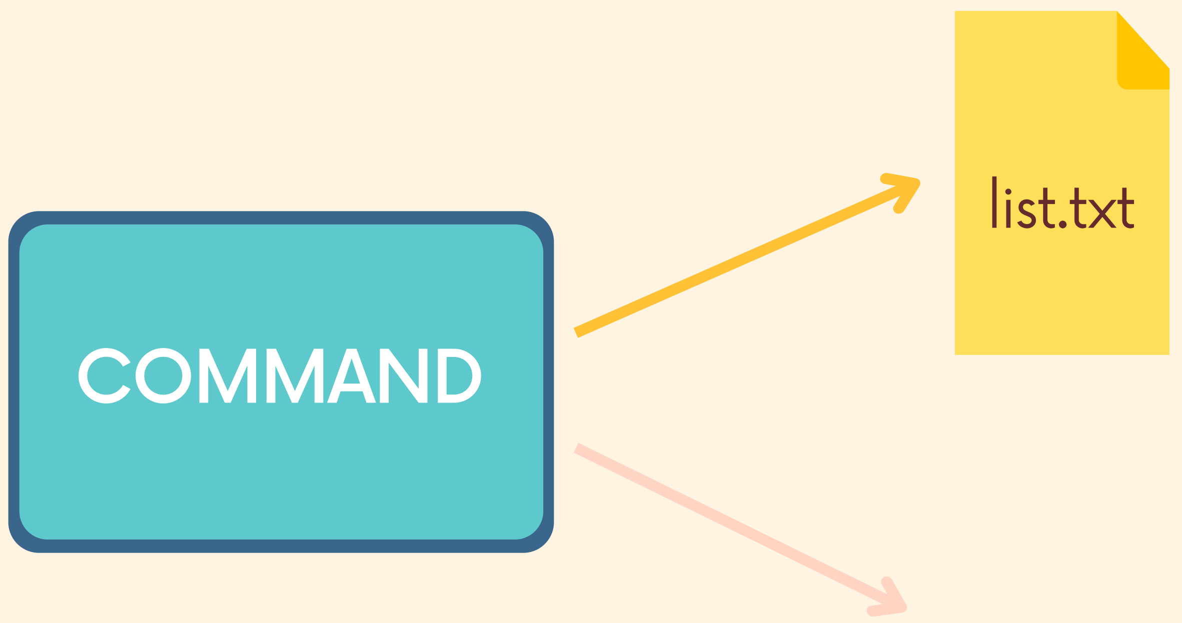
| connects a command to another command.

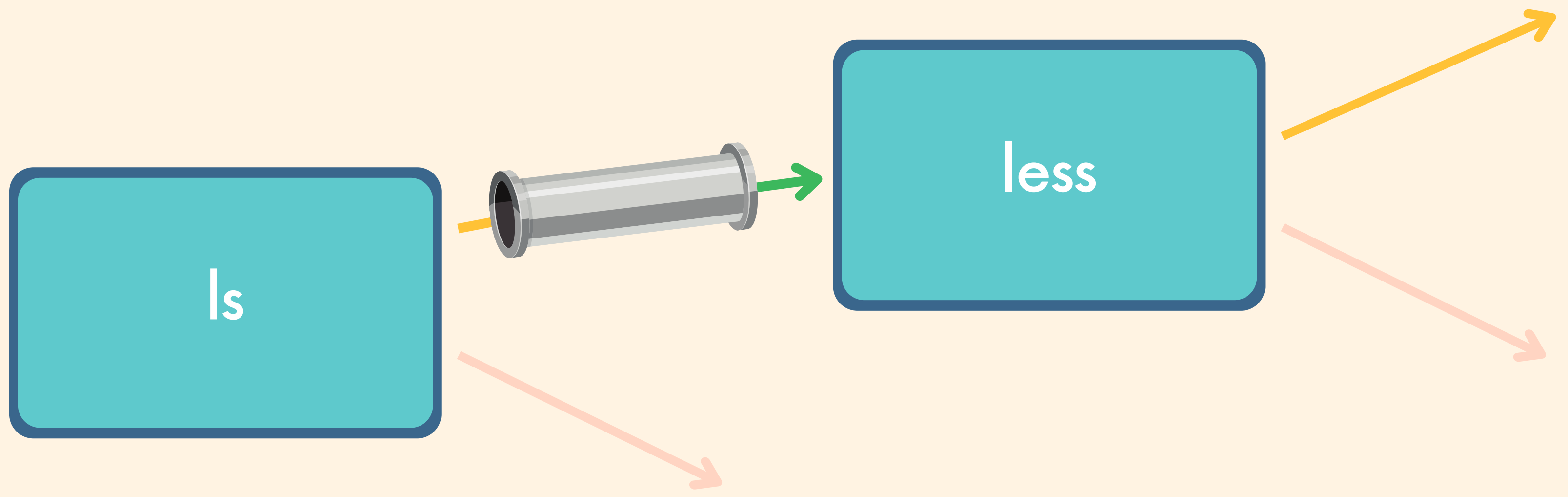


```
ls -l /usr/bin > list.txt
```



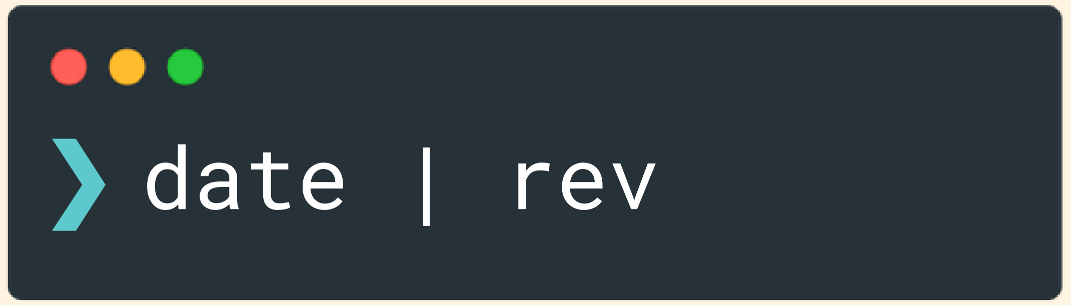
```
ls -l /usr/bin | less
```



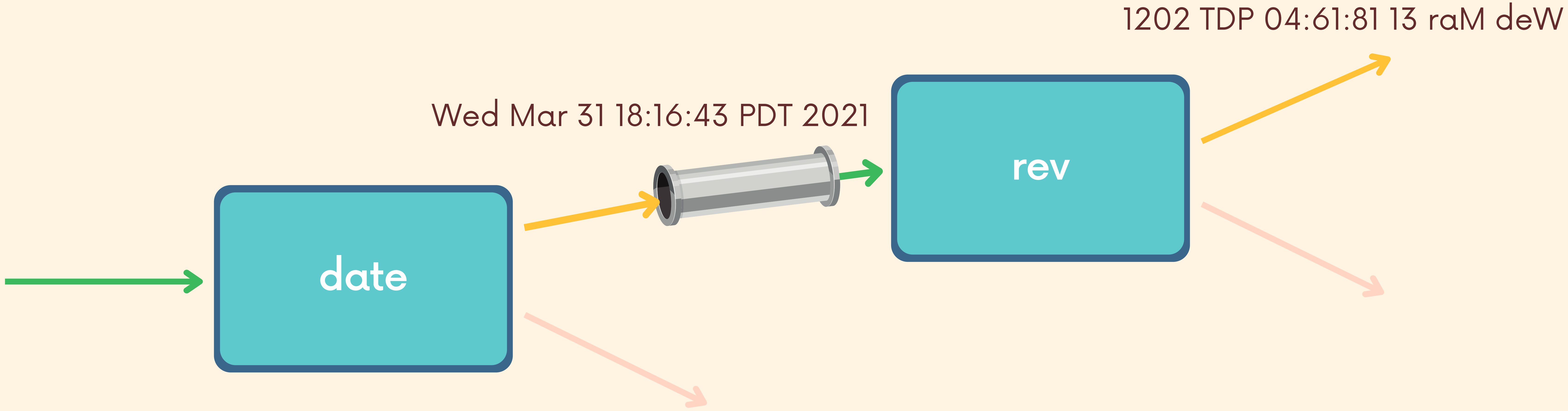


date | rev

This example shows the output of the date command being piped to the rev command. The end result is the reverse of the current date! Very useful!

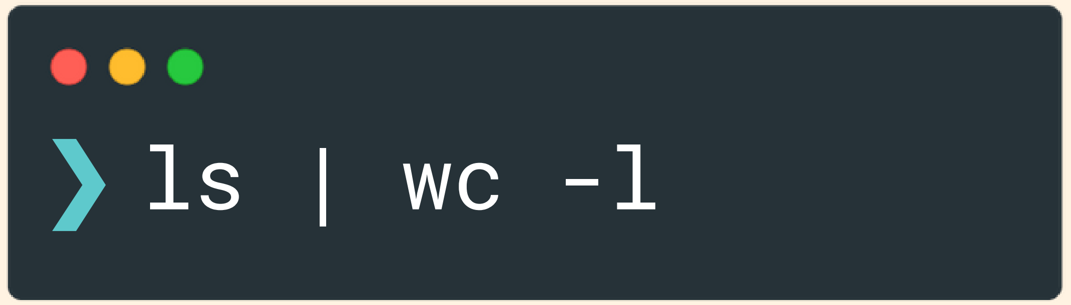


```
❯ date | rev
```

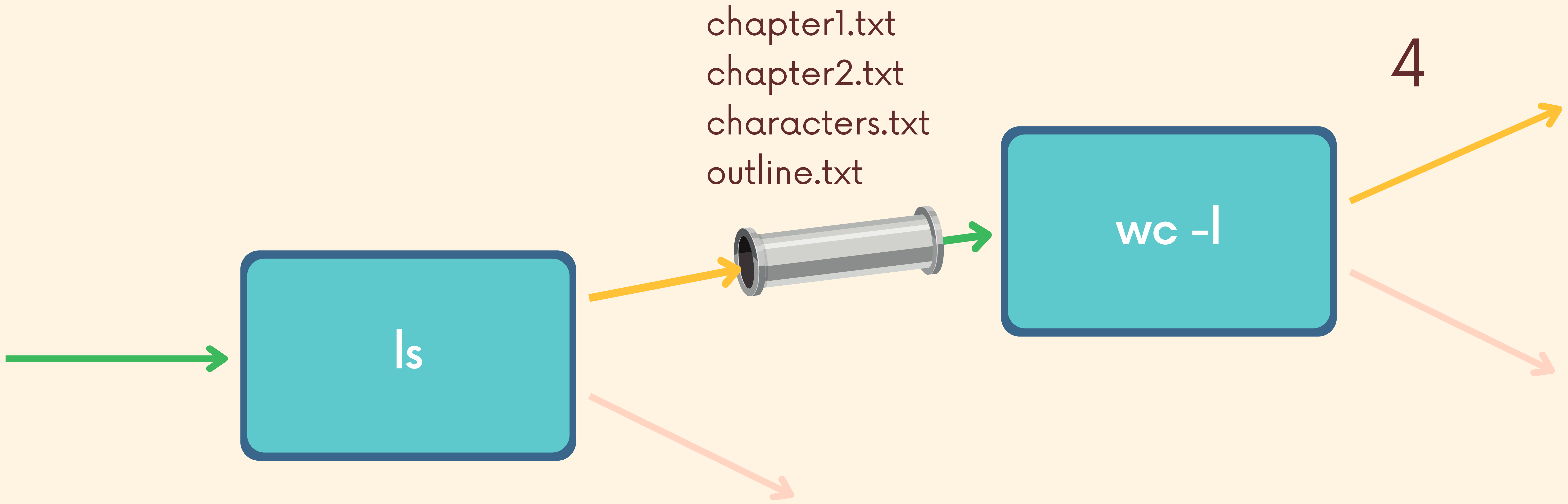


ls | wc

This example counts the number of files (non hidden files) in a directory. We pipe the output of ls to the word count command. The -l option tells wc to count the number of lines.

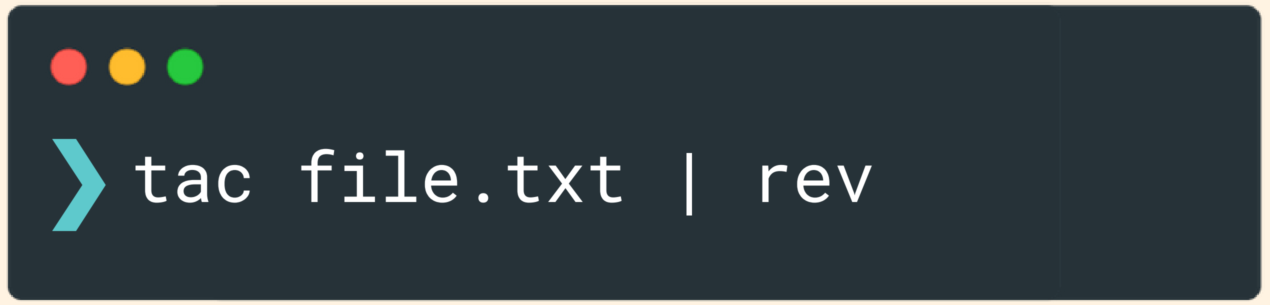


```
❯ ls | wc -l
```

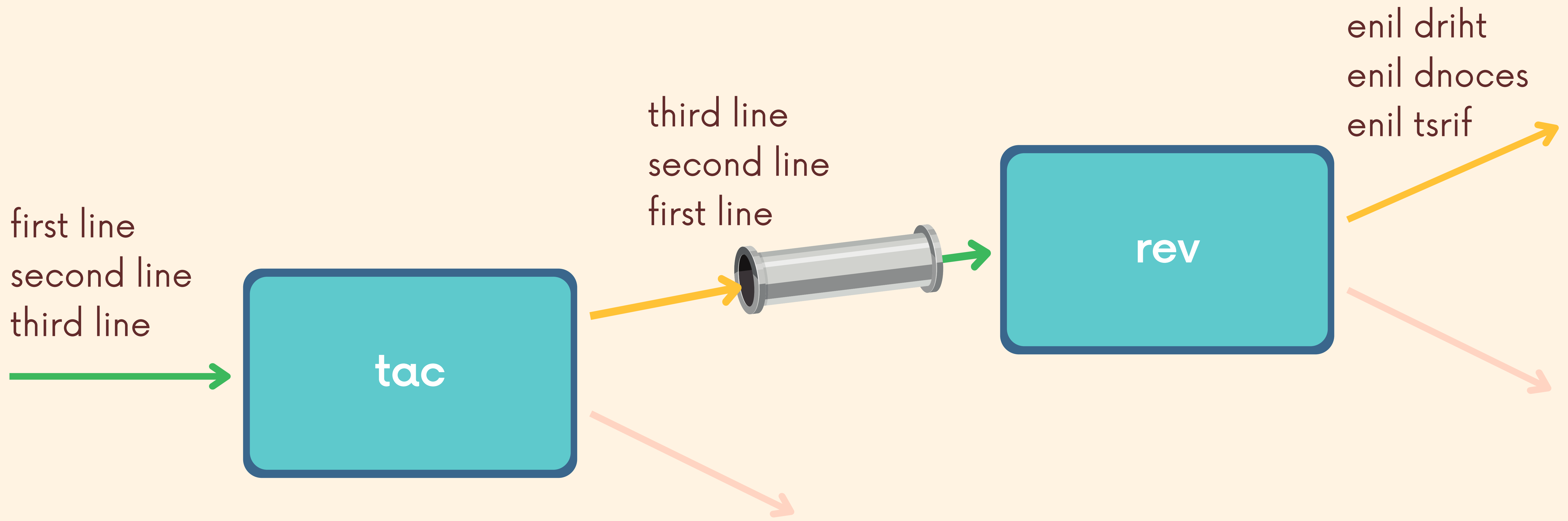


tac | rev

In this example, we are calling tac with a file and then piping the output to rev. The final result is the content of file.txt printed "horizontally" and "vertically" reversed

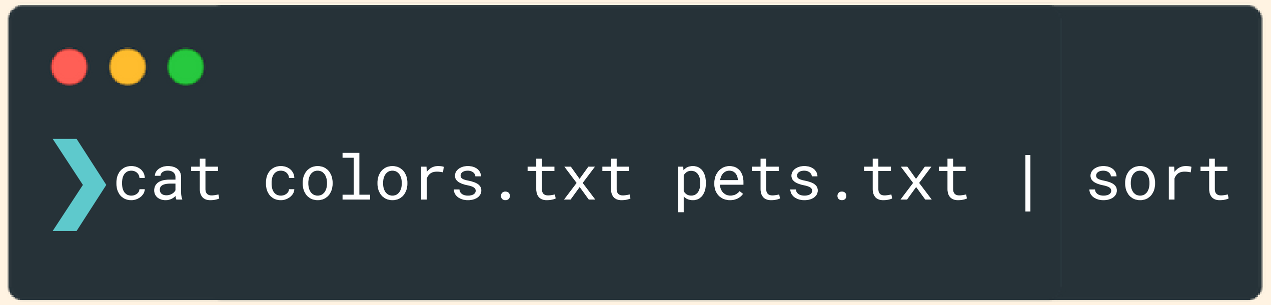


```
❯ tac file.txt | rev
```

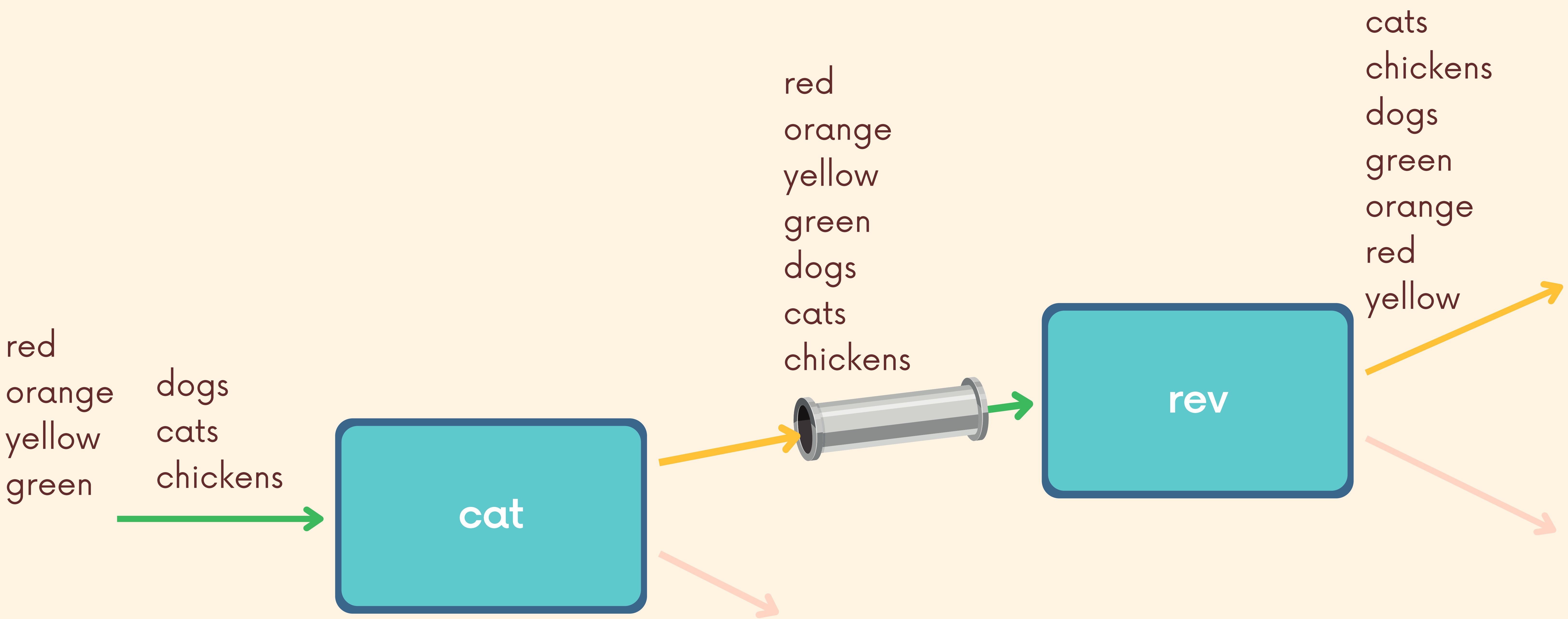


cat | sort

This example concatenates two files using cat and then sorts them alphabetically.

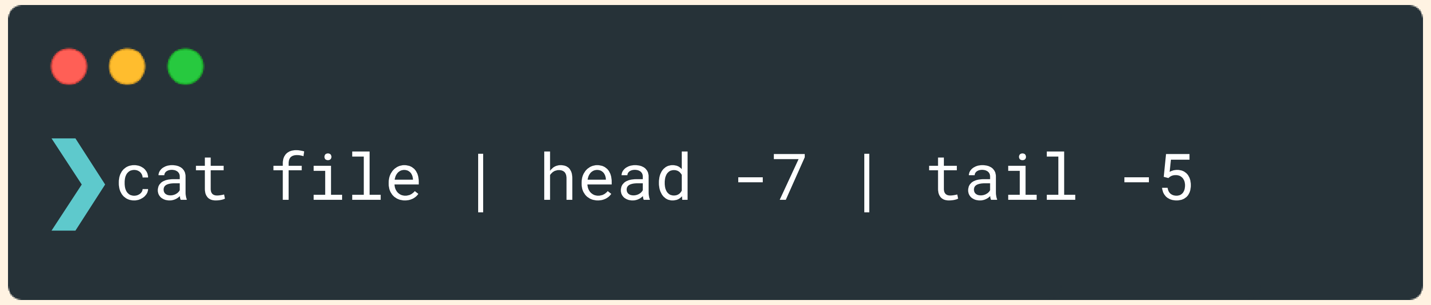


```
❯cat colors.txt pets.txt | sort
```

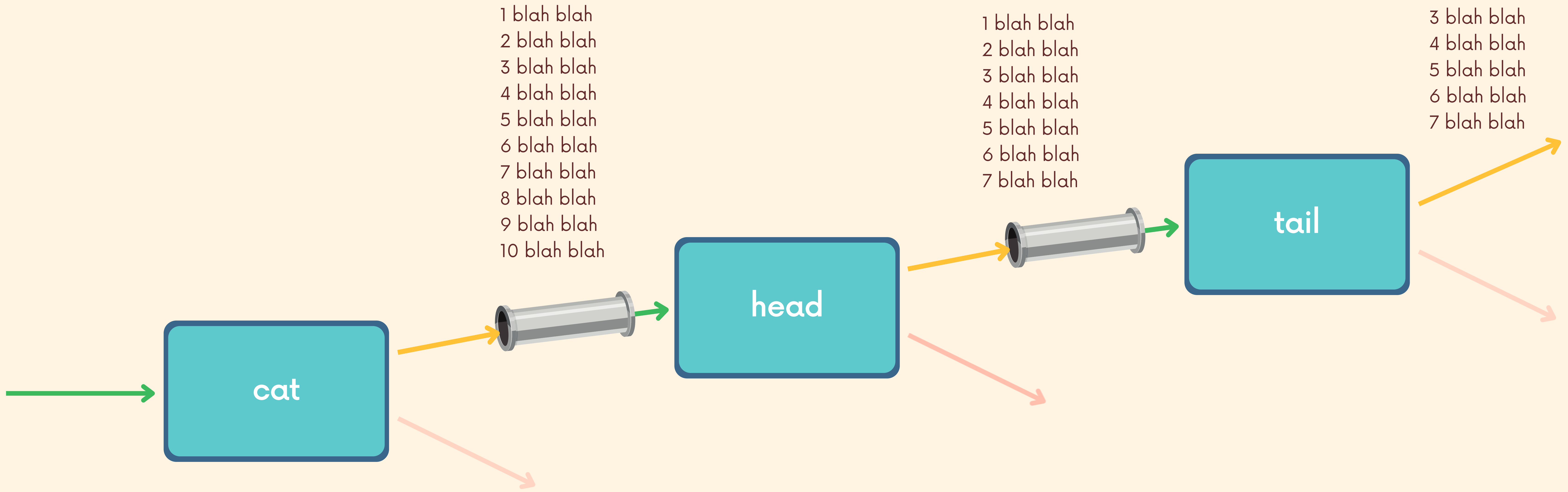


cat | head | tail

In this example, we are using cat to feed a file to head, which cuts it down to the first 7 lines of the file and passes it to tail, which then outputs the last 5 lines of that "chunk"
The end result is lines 3-7 are output to the screen

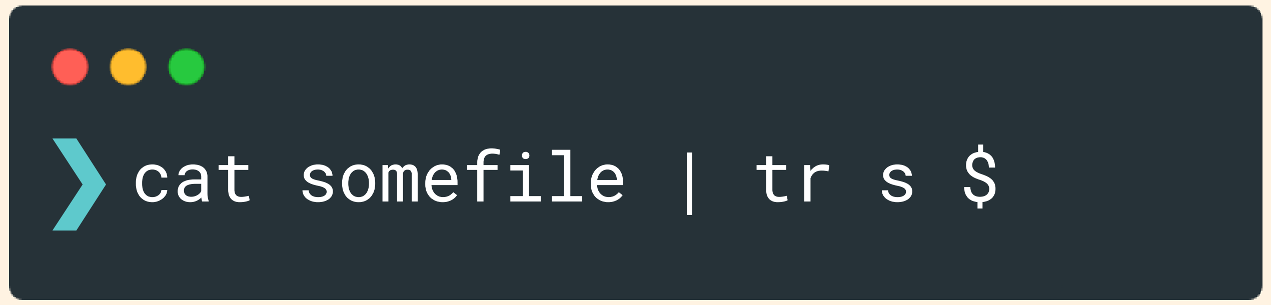


```
❯cat file | head -7 | tail -5
```

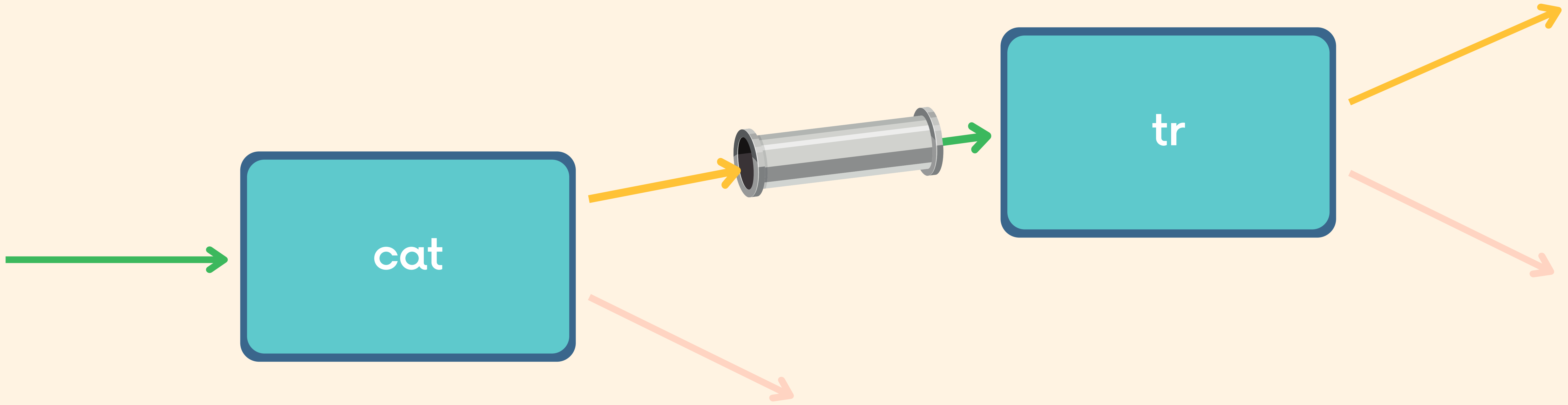


The Syntax

We use the pipe character (|) to separate two commands. The output of the first command will be passed to the standard input of the second command.

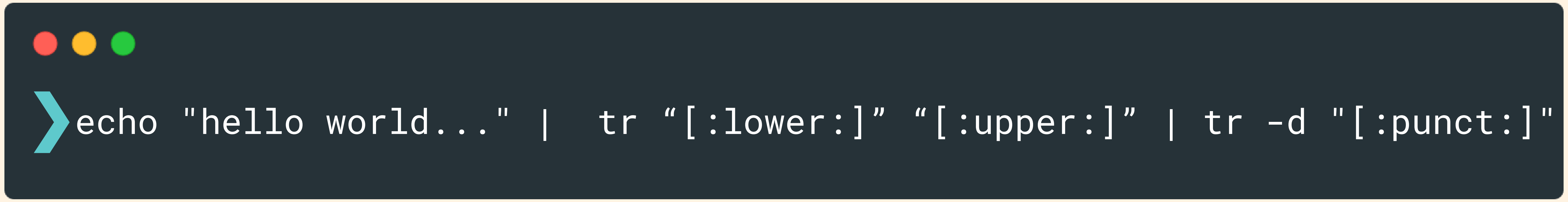


```
❯ cat somefile | tr s $
```

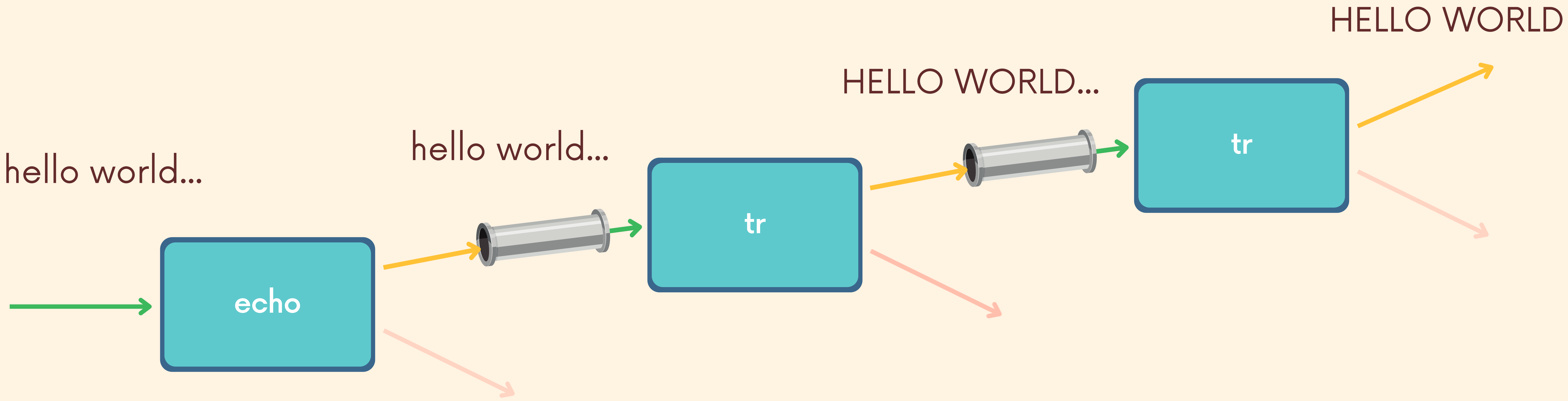


echo | tr | tr

This example uses **tr** to capitalize a string and then again uses **tr** to remove all punctuation from the capitalized string.



```
❯echo "hello world..." |  tr “[:lower:]” “[:upper:]” | tr -d "[:punct:]"
```



ls | sort | head

```
❯ls -lh | sort -rhk 5 | head -3
```

This example displays the 3 largest files in the current directory, using ls, sort, and head.

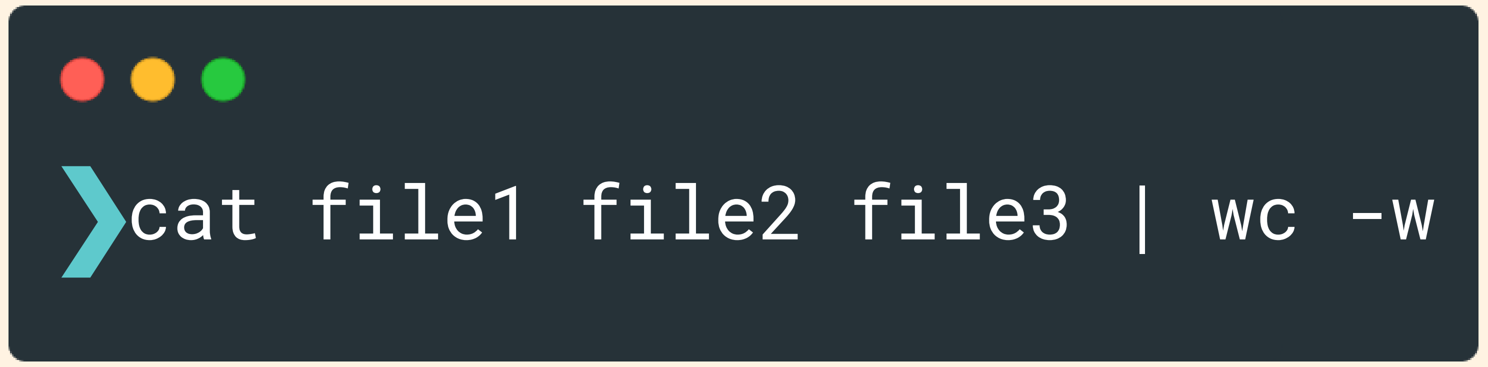
First, ls -lh lists out all the files in the current directory. That output is passed to sort, which sorts based on the fifth field (the file size). The -h option is for human readable sort (comparing 100b, 40k, 1g, etc), and the -r reverses the order so that we end up with the largest files first.

Finally, that output is passed to head, which limits the results to the first 3.

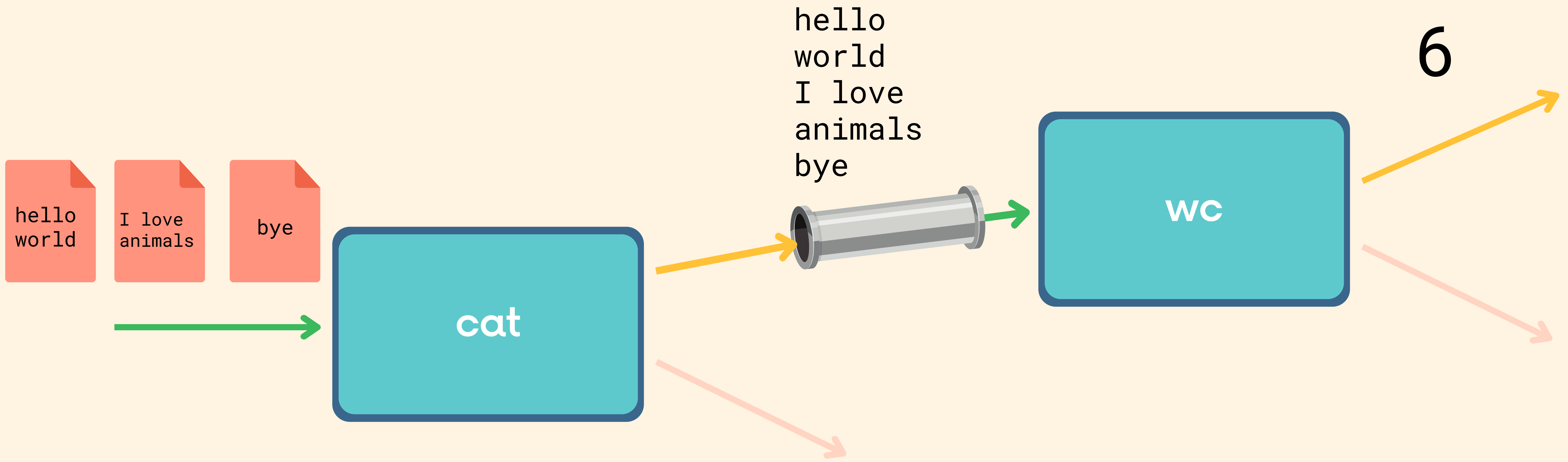
*NOTE: this is not the preferred way to find large files! Use the du command instead

An Example

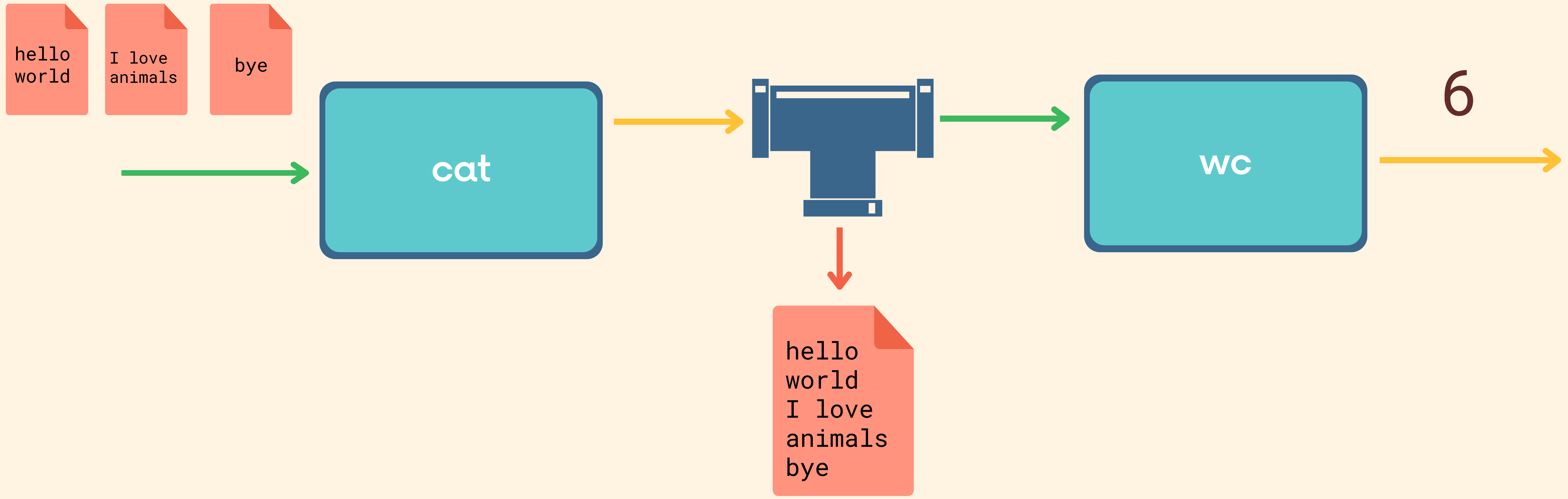
In this example, I'm using cat to concatenate three files together before piping the output to wc to get a count of the total number of words.



```
❯cat file1 file2 file3 | wc -w
```

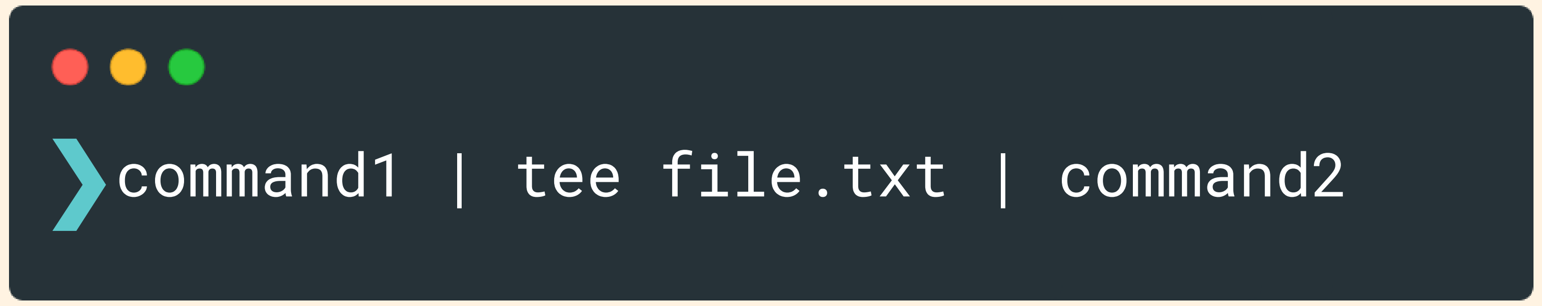


What if I wanted to create a file with the output of cat?

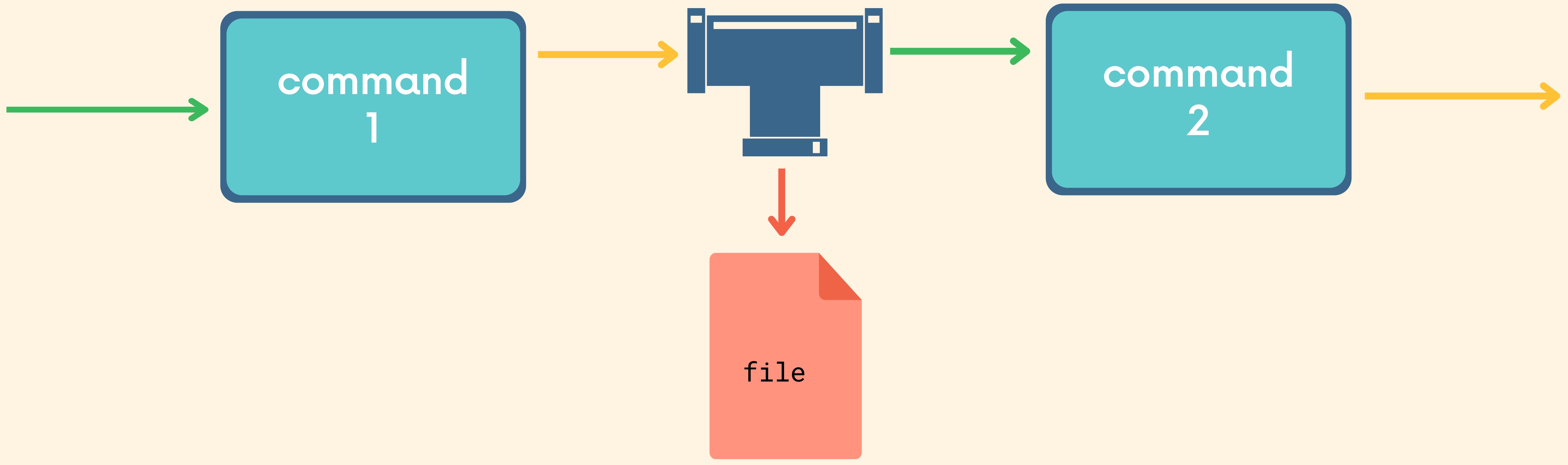


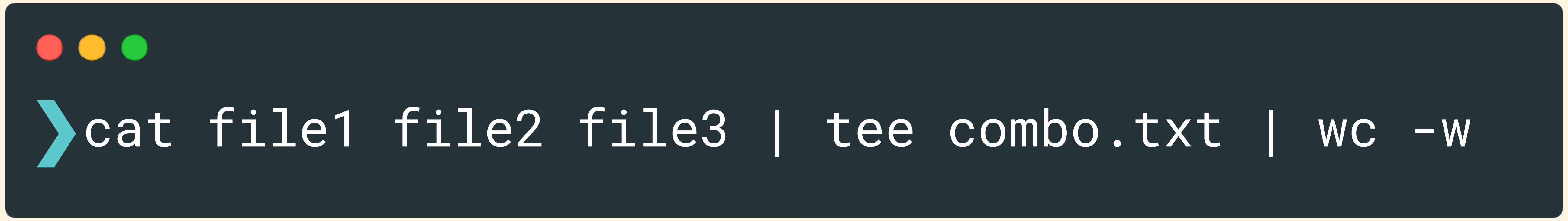
enter the tee command

The tee program reads standard input and copies it both to standard output AND to a file. This allows us to capture information part of the way through a pipeline, without interrupting the flow.

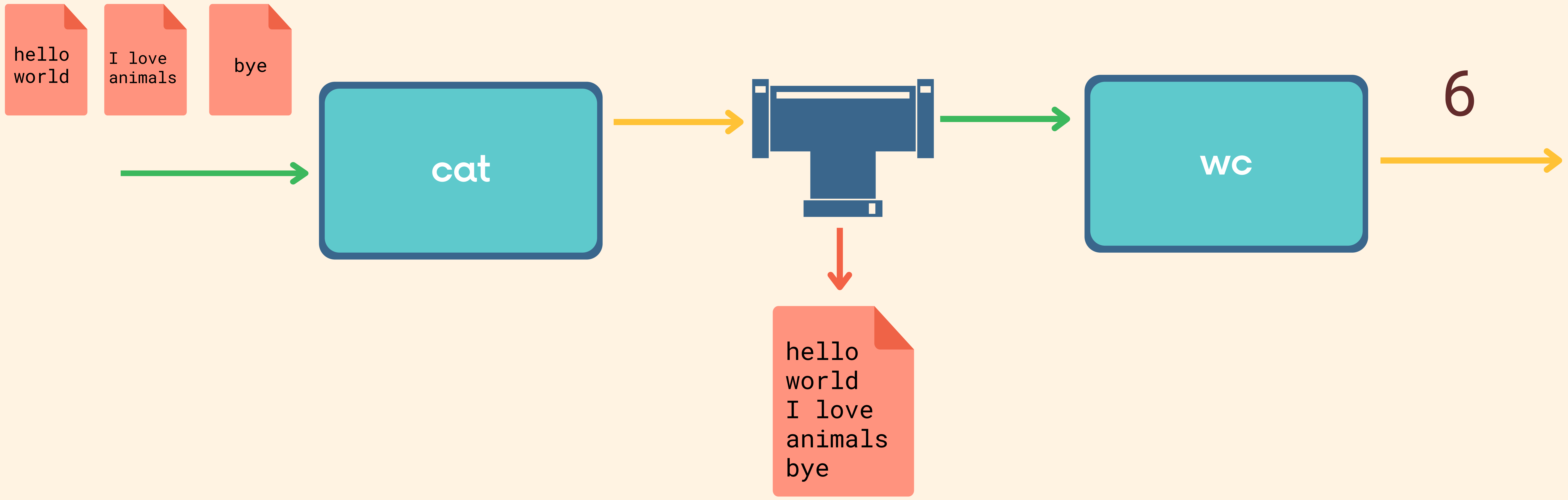


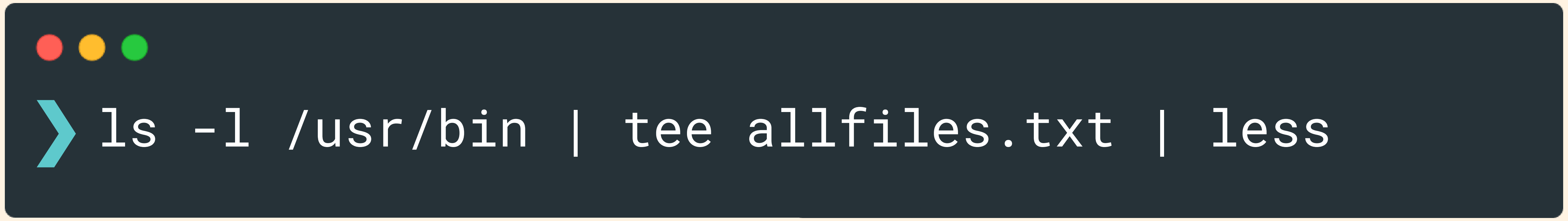
```
❯command1 | tee file.txt | command2
```



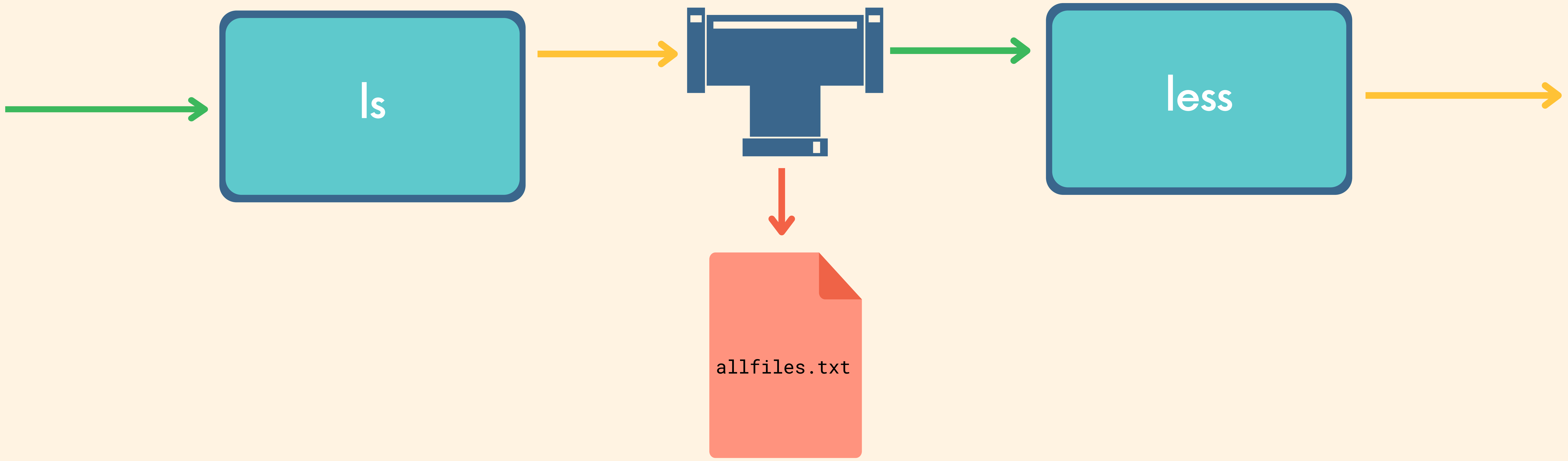


```
❯cat file1 file2 file3 | tee combo.txt | wc -w
```





```
❯ ls -l /usr/bin | tee allfiles.txt | less
```


ls
less
allfiles.txt