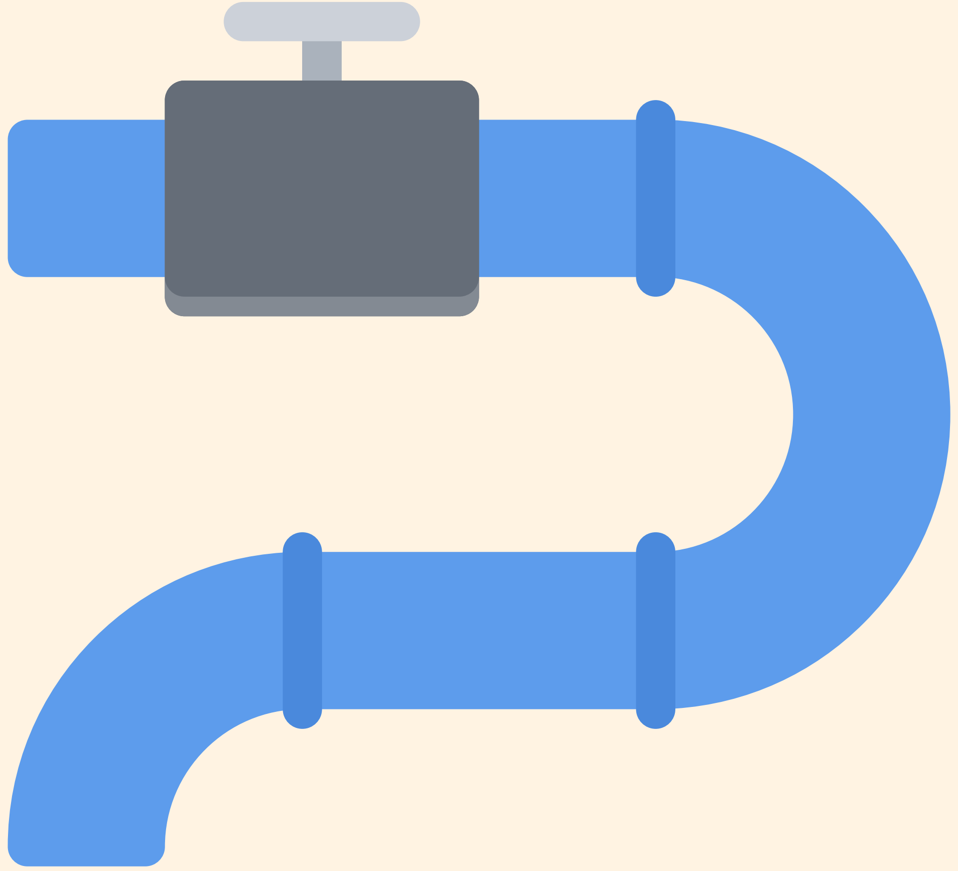
Redirection

Standard Streams

The three standard streams are **communication channels** between a computer program and its environment.

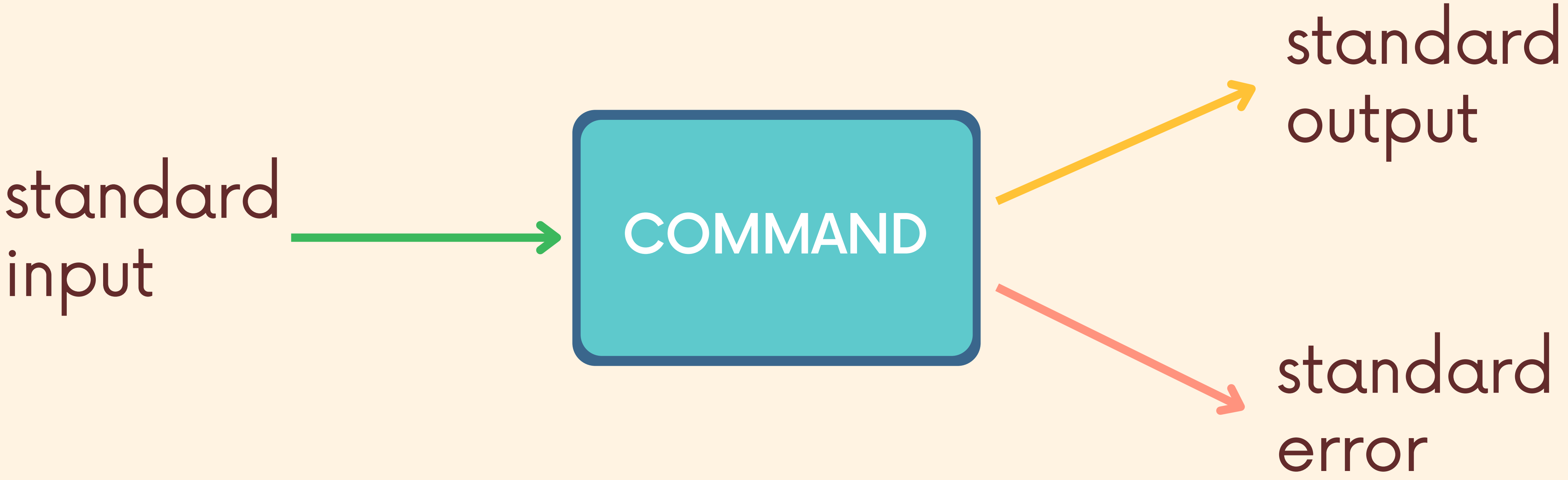
They are:
- Standard Input
- Standard Output
- Standard Error

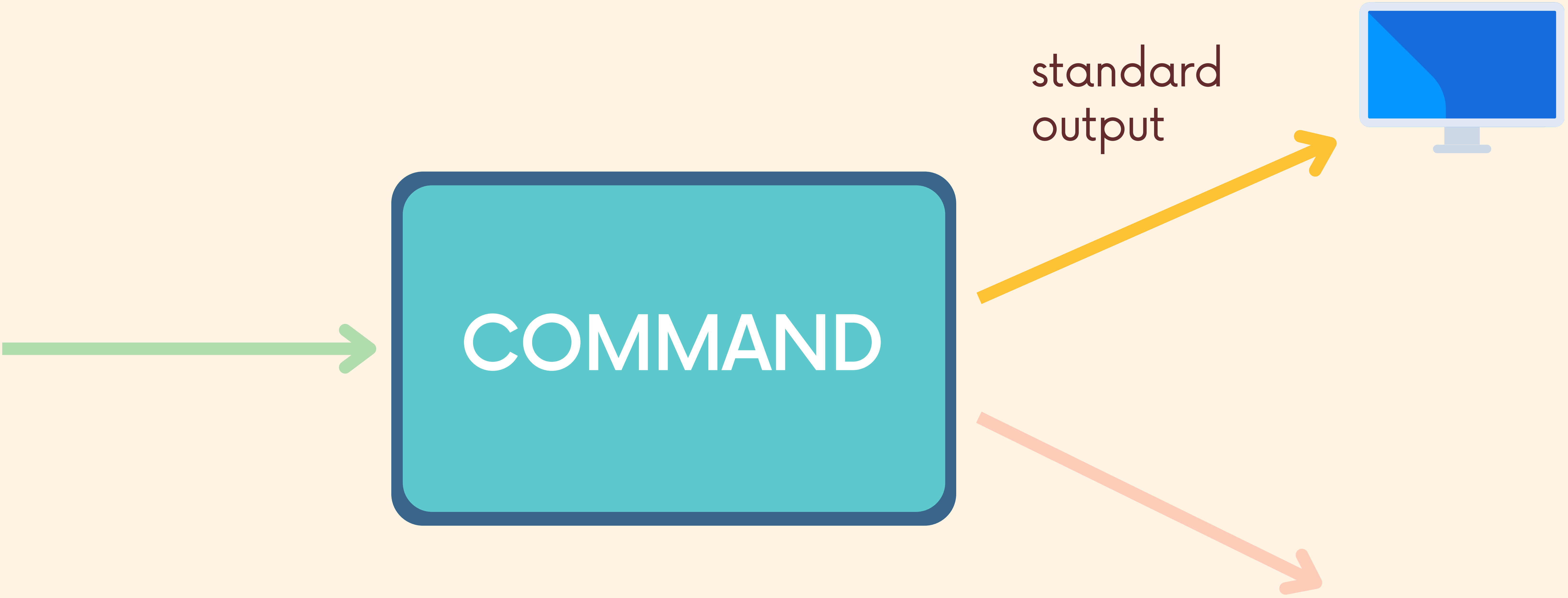


Standard Output

Standard output is a place to which a program or command can send information.

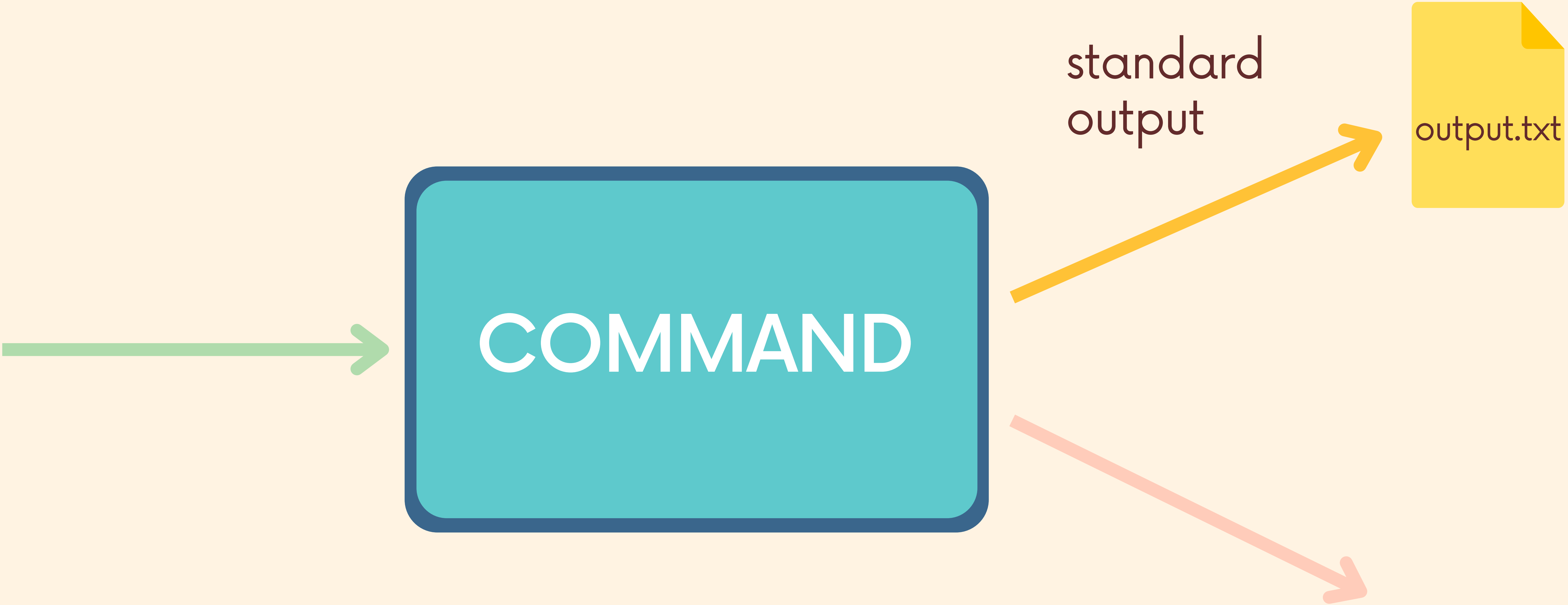
The information could go to a screen to be displayed, to a file, or even to a printer or other devices.





standard
output

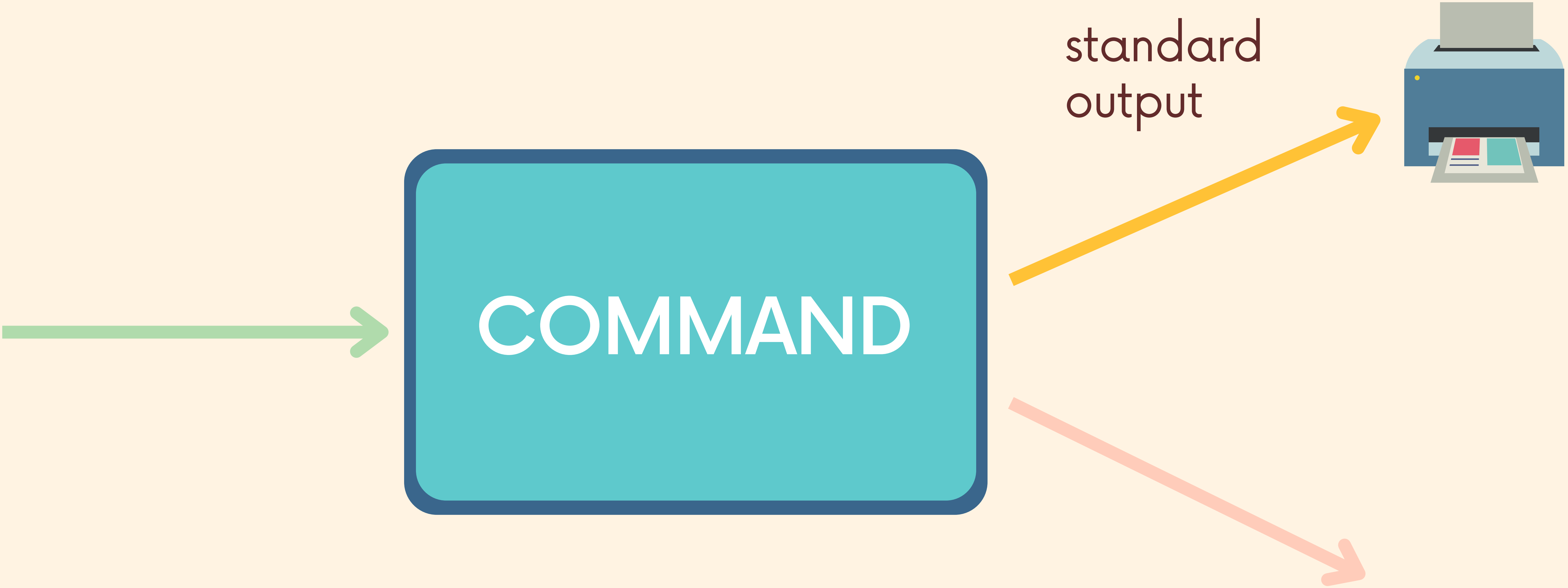
COMMAND



COMMAND

standard
output

output.txt



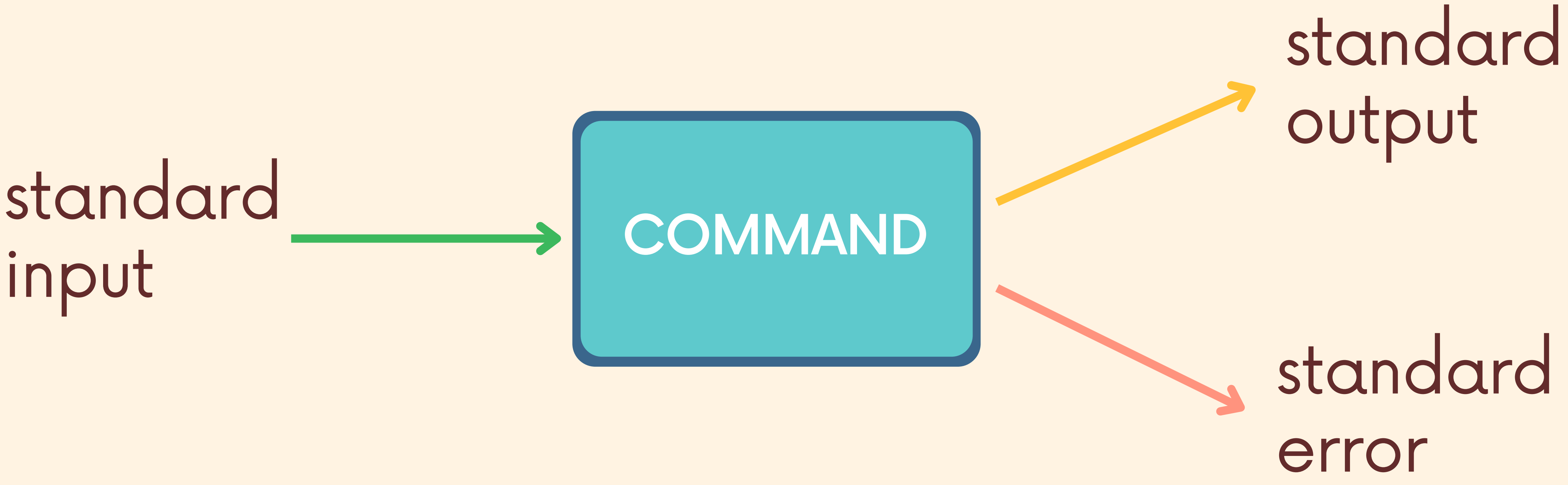
COMMAND

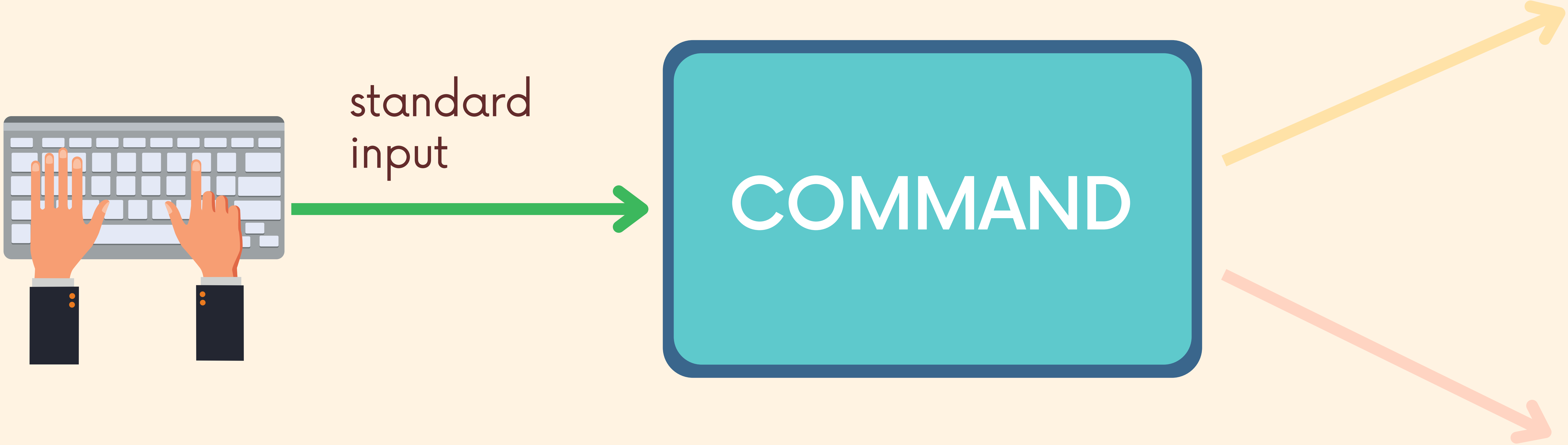
standard output

Standard Input

Standard input is where a program or command gets its input information from. By default, the shell directs standard input from the keyboard.

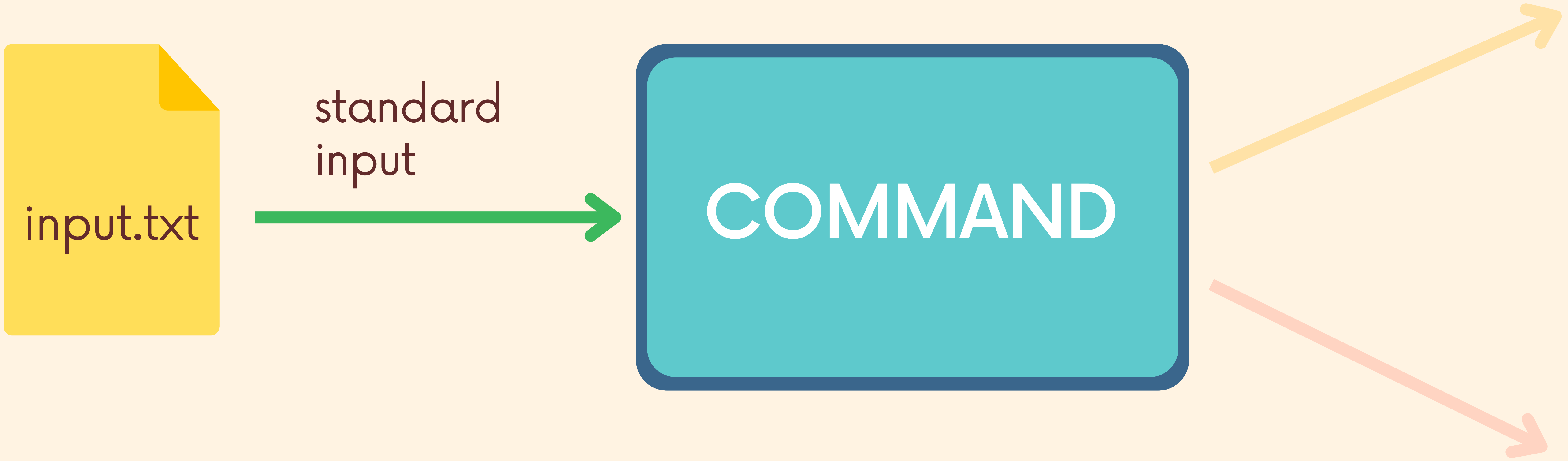
The input information could come from a keyboard, a file, or even from another command!

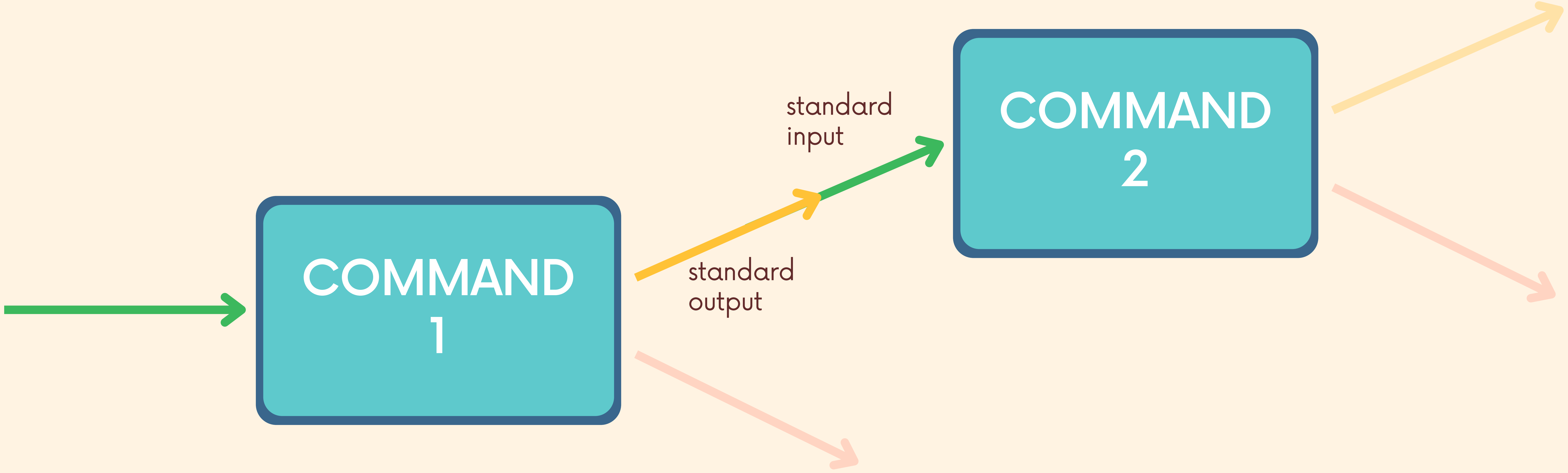




standard input

COMMAND

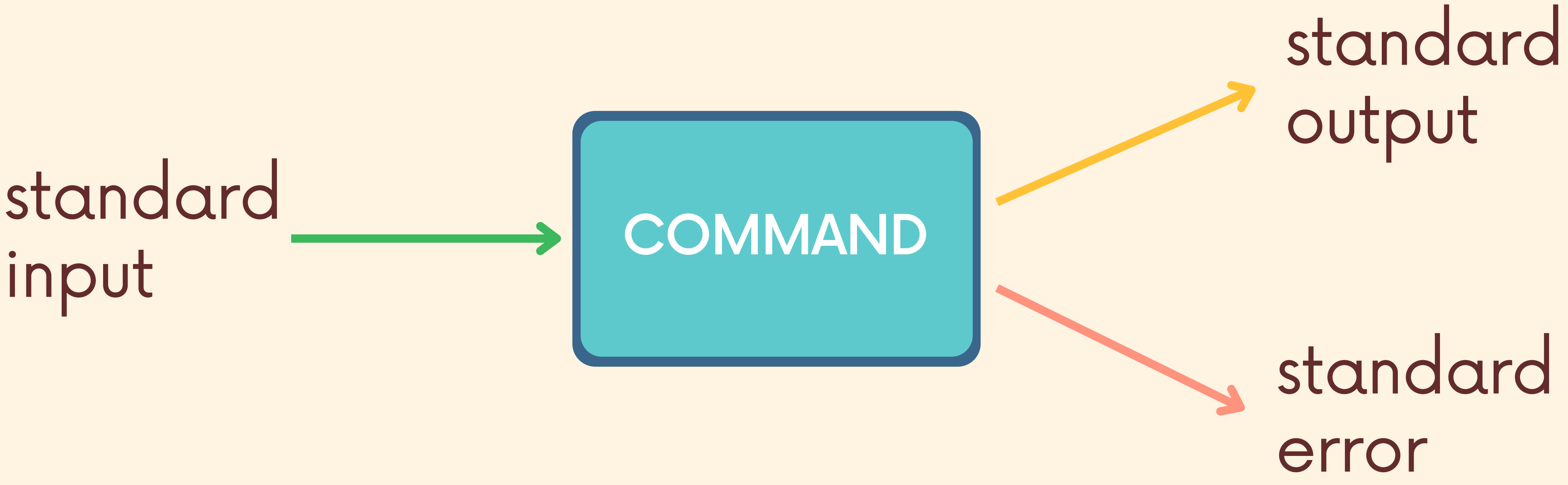
input.txt
standard input
COMMAND


standard
input
COMMAND
2
COMMAND
1
standard
output

Standard Error

Commands and programs also have a destination to send error messages: standard error.

By default, the shell directs standard error information to the screen for us to read, but we can change that destination if needed!



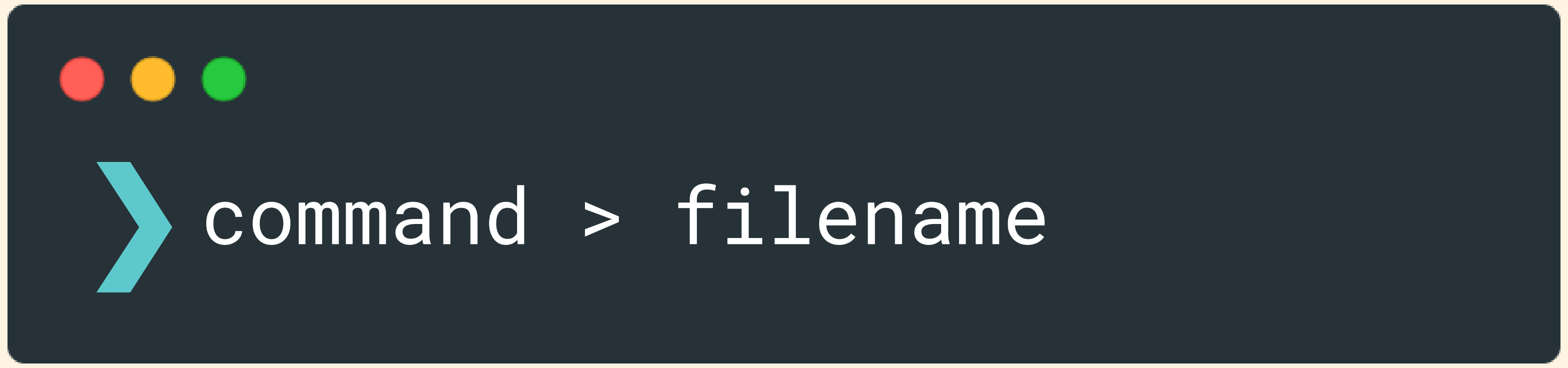
redirection

"redirection" describes the ways we can alter the source of standard input, and the destinations for standard output and standard error.

redirecting output

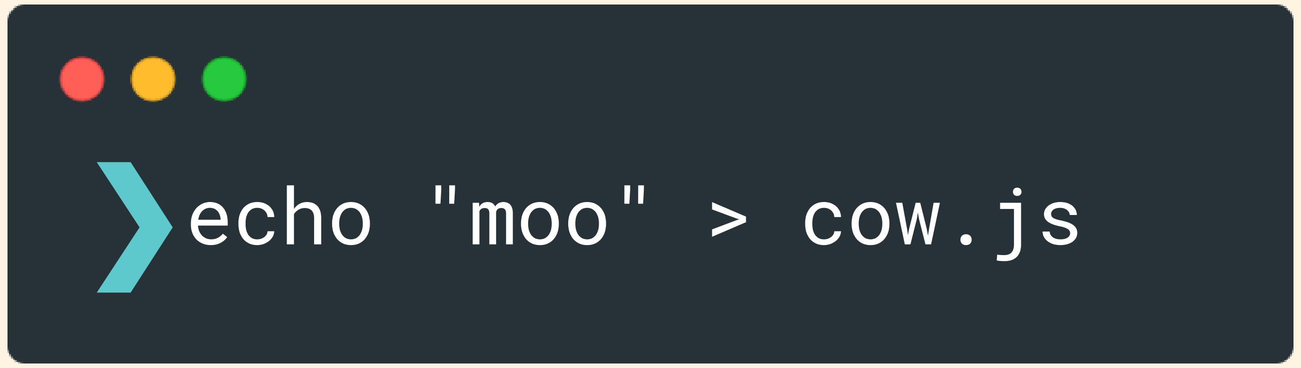
The redirect output symbol (>) tells the shell to redirect the output of a command to a specific file instead of the screen.

By default, the date command will print the current date to the screen. If we instead run date > output.txt the output will be redirected to a file called output.txt



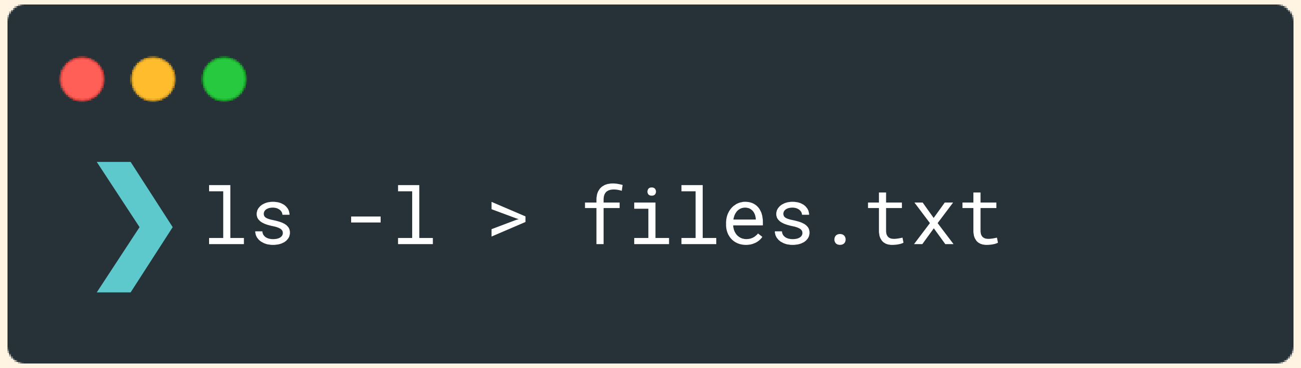
```
❯ command > filename
```

redirecting output



```
echo "moo" > cow.js
```

This example redirects the output of echo



```
ls -l > files.txt
```

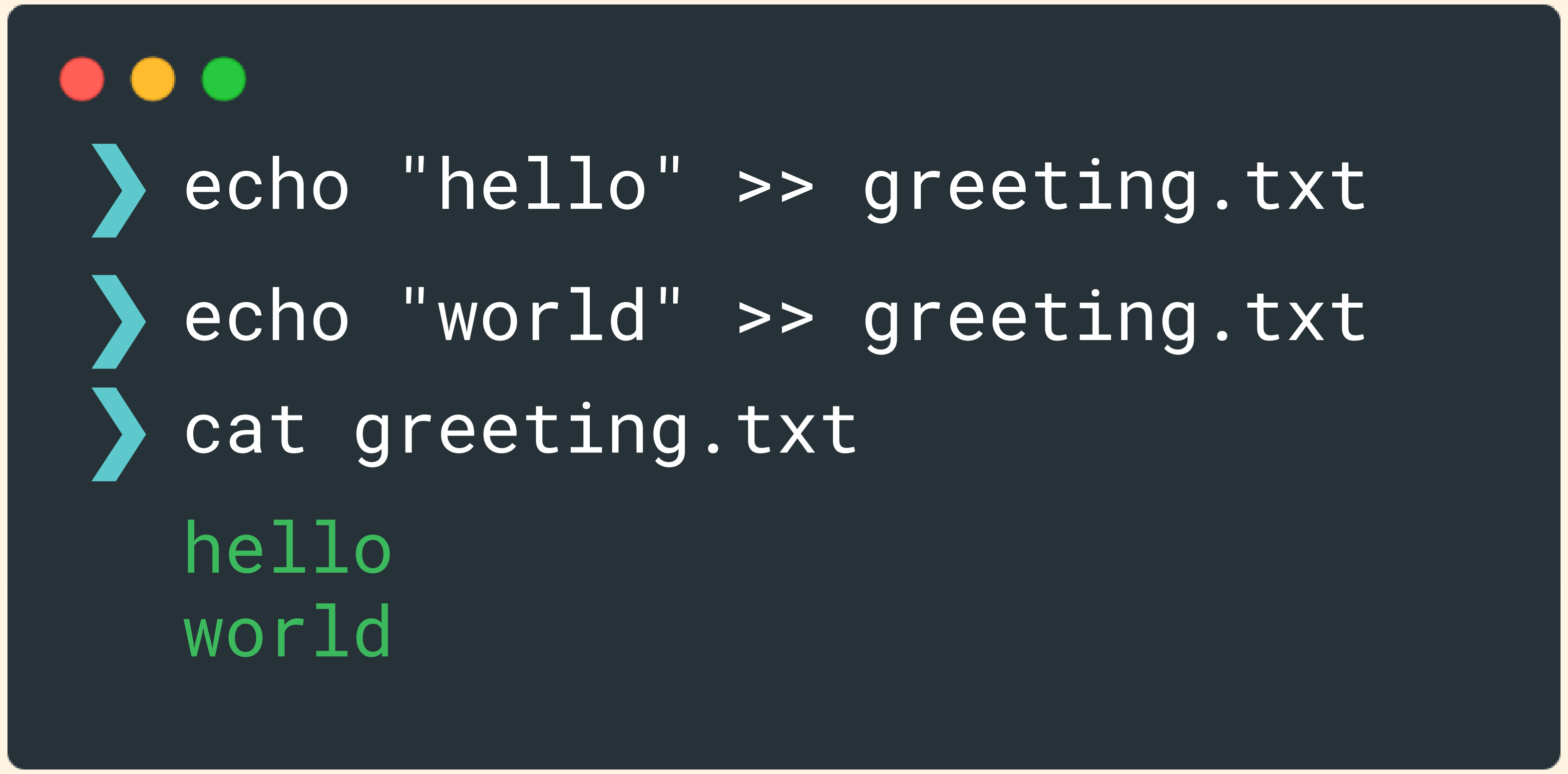
This example saves the output of ls -l to a file.

*Note the > symbol needs to occur after any options and arguments!

Appending

When we redirect output into a file using > any existing contents in the file are overwritten. Sometimes this is not what we want!

To instead keep the existing contents in the file and add new content to the end of the file, use >> when redirecting.



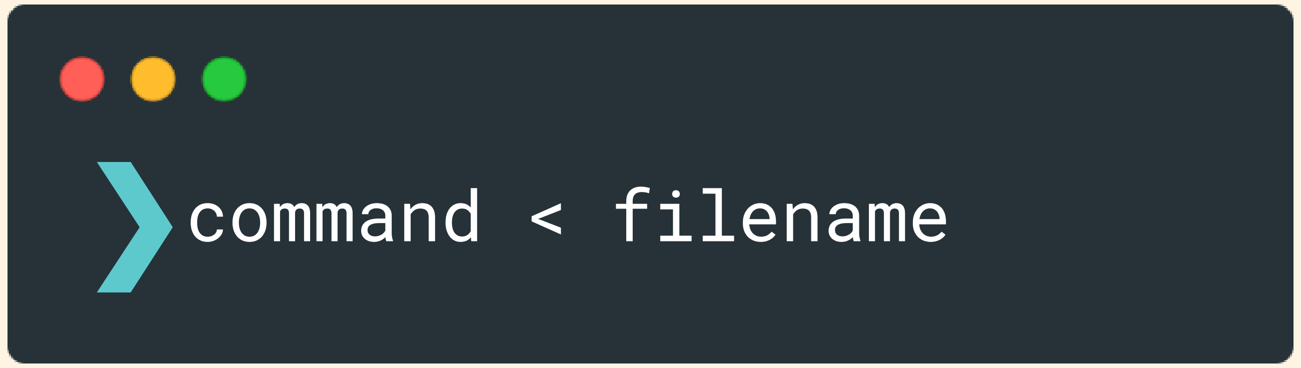
```
❯ echo "hello" >> greeting.txt
❯ echo "world" >> greeting.txt
❯ cat greeting.txt
  hello
  world
```

redirecting input

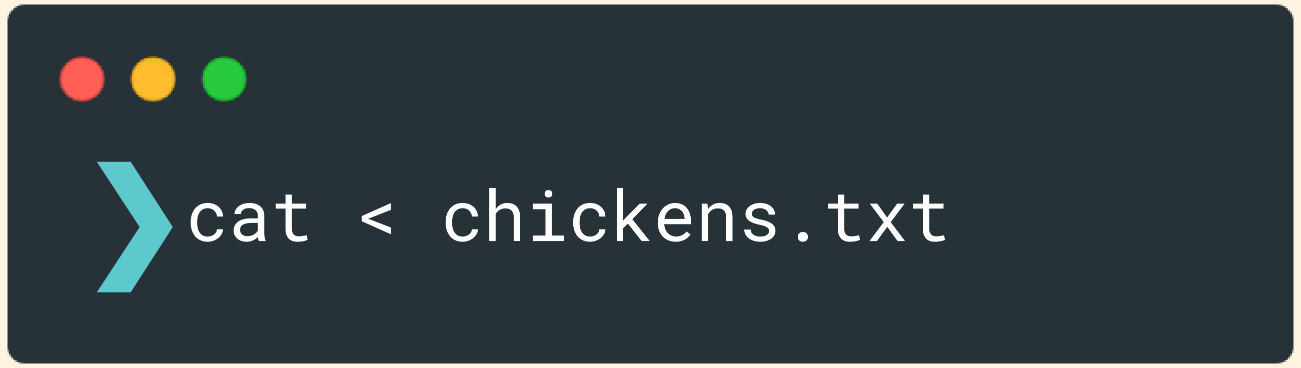
To pass the contents of a file to standard input, use the < symbol followed by the filename.

For example, we could pass the contents of the chickens.txt file to the cat command using
cat < chickens.txt

cat (and many other commands) are set up to accept filenames as arguments directly, but we can also redirect to standard input manually.



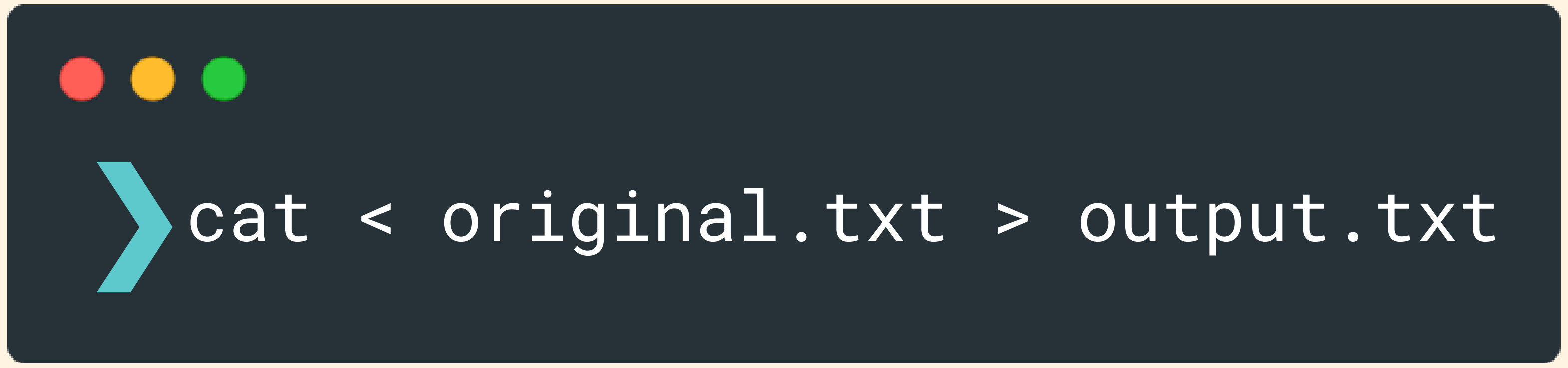
```
command < filename
```



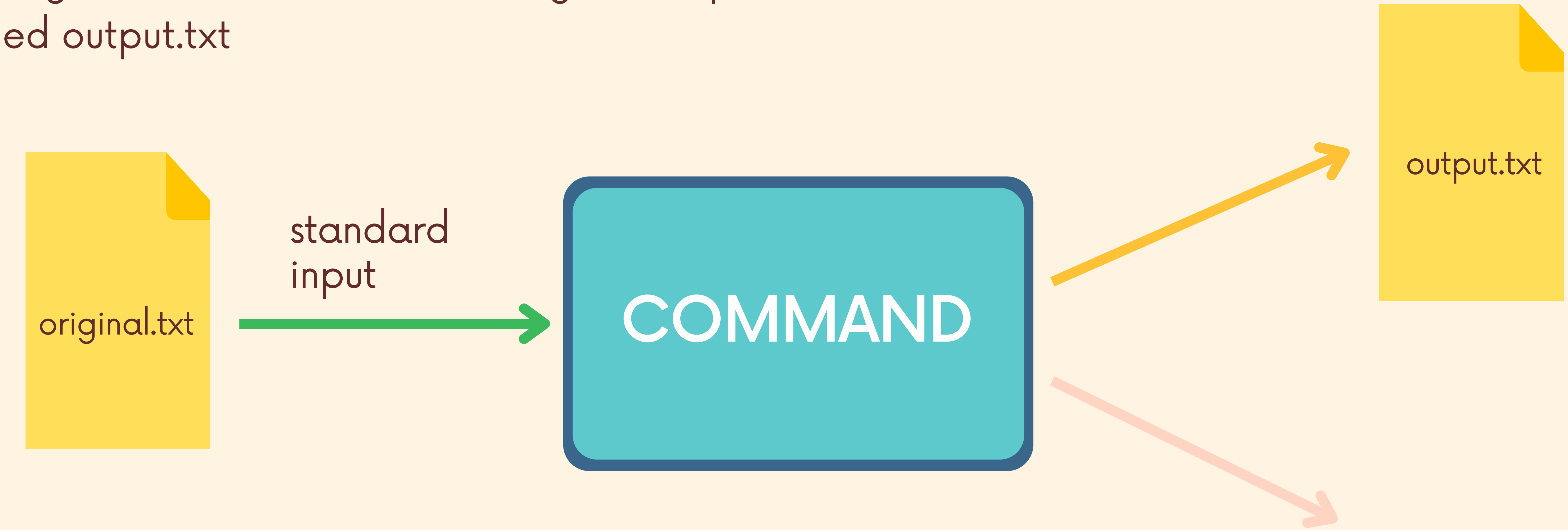
```
cat < chickens.txt
```

combo!

We can redirect standard input and output at the same time! In this example, we are using cat to read in the contents of original.txt and then redirecting the output to a file called output.txt

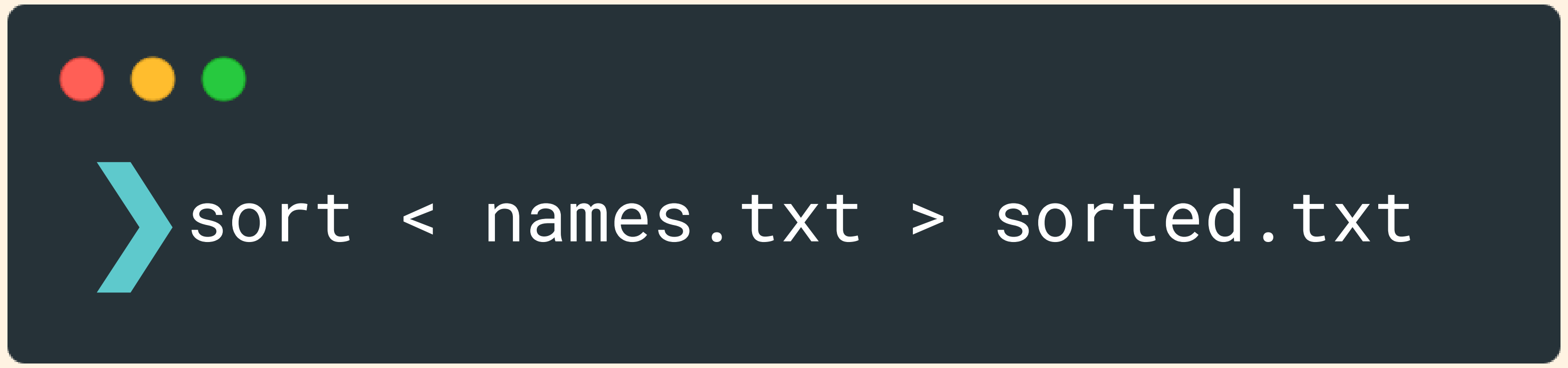


```
cat < original.txt > output.txt
```

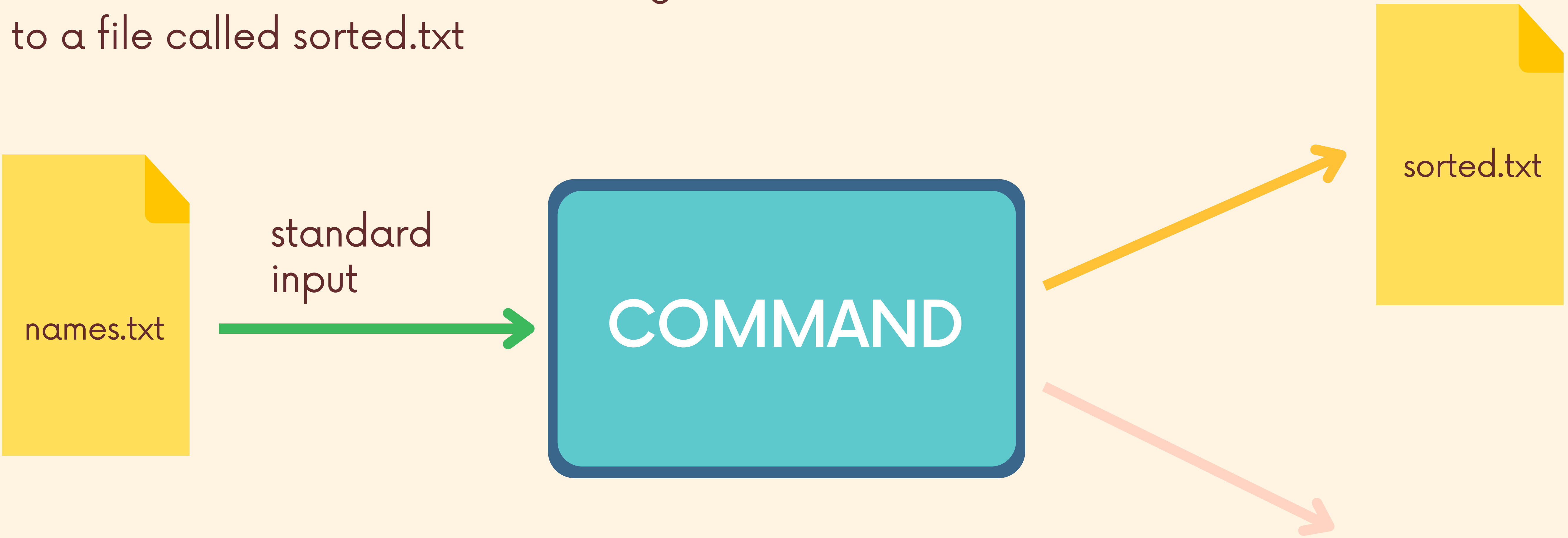


combo! again!

We can redirect standard input and output at the same time! In this example, we are redirecting the names.txt file to the sort command. We are also redirecting the output of sort to a file called sorted.txt



```
sort < names.txt > sorted.txt
```

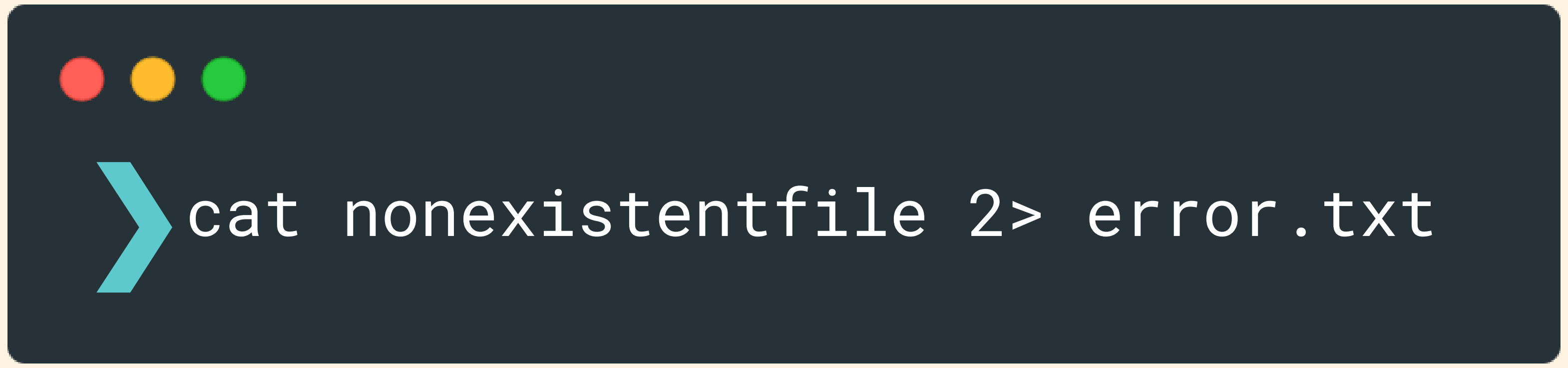


redirecting standard error

By default, error messages are output to the screen, but we can change this by redirecting standard error.

The standard error redirection operator is 2>

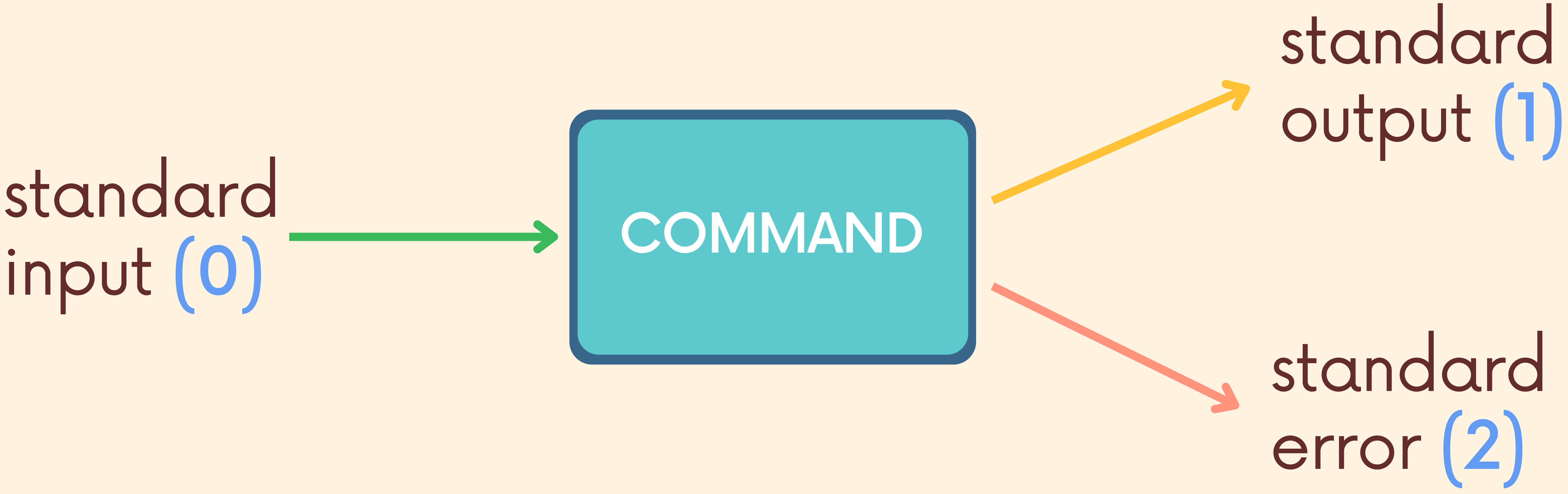
If we ran a command like cat nonexistentfile (where the file really does not exist) we would see an error printed to the screen. We can instead redirect standard error to a file with cat nonexistentfile 2> error.txt



```
cat nonexistentfile 2> error.txt
```

Why 2> ?

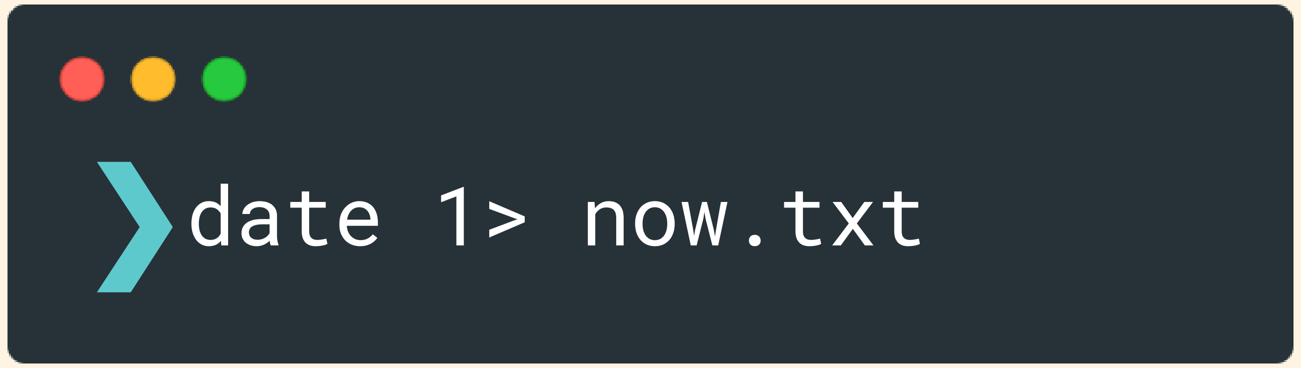
Each stream gets its own numeric file descriptor, and for standard error the number is 2.



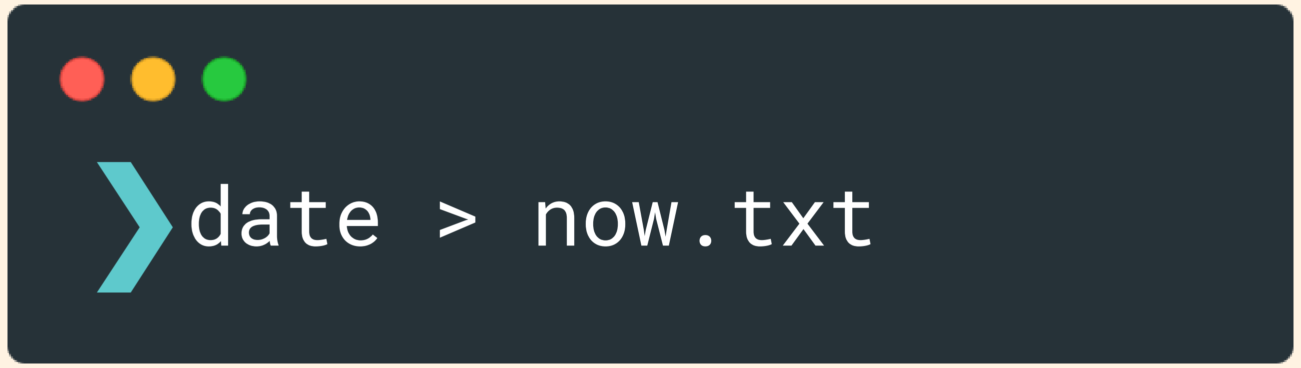
defaults

The > operator actually defaults to using 1 as the file descriptor number, which is why we didn't need to specify 1> to redirect standard output

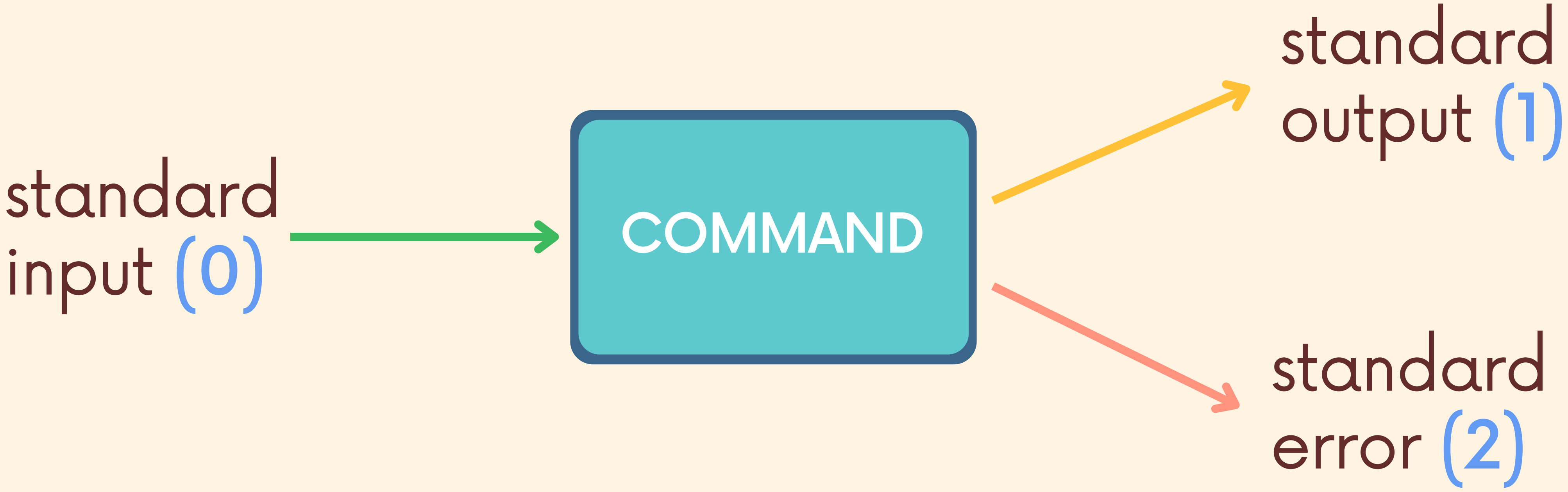
Similarly, the < operator uses a default file descriptor number of 0, so we don't need to specify 0< to redirect to standard input (though we can!)



```
❯date 1> now.txt
```



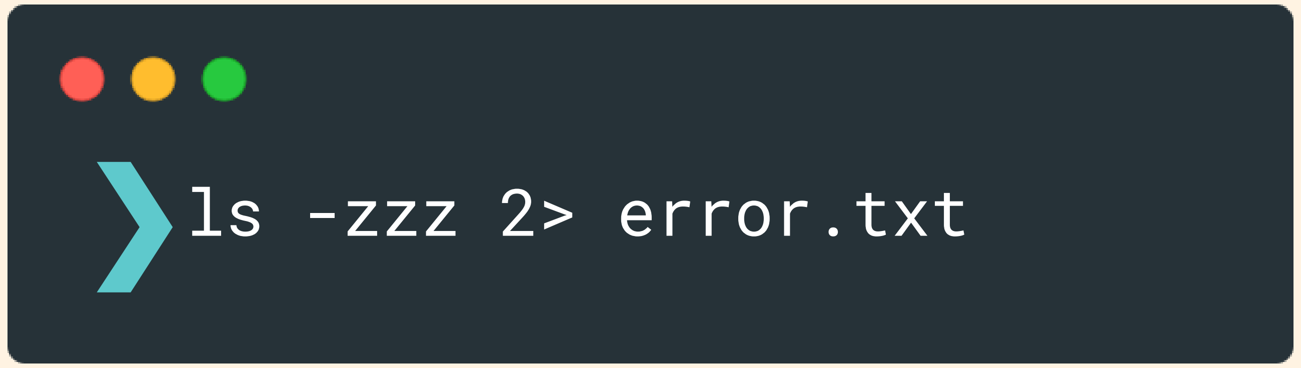
```
❯date > now.txt
```



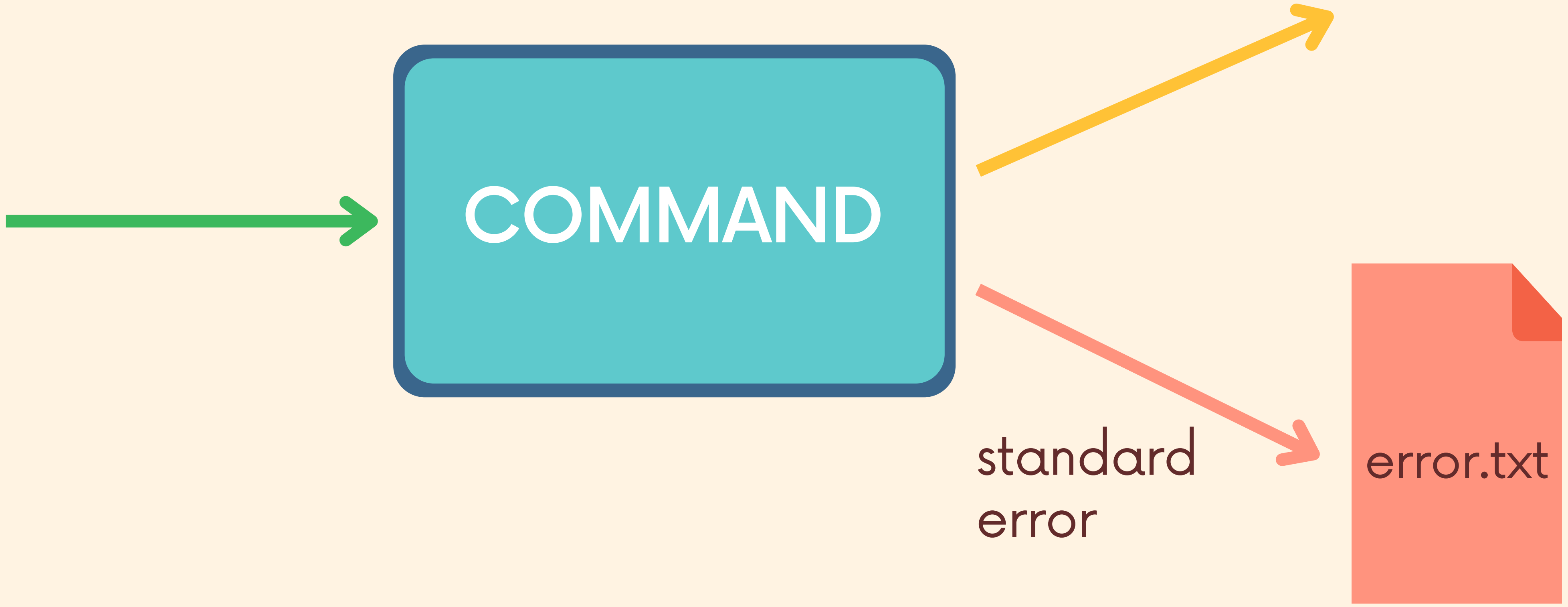
another one

In this example, I'm running **ls** with an invalid option that generates the error message "ls: illegal option -- z"

I'm redirecting standard error to an error.txt file

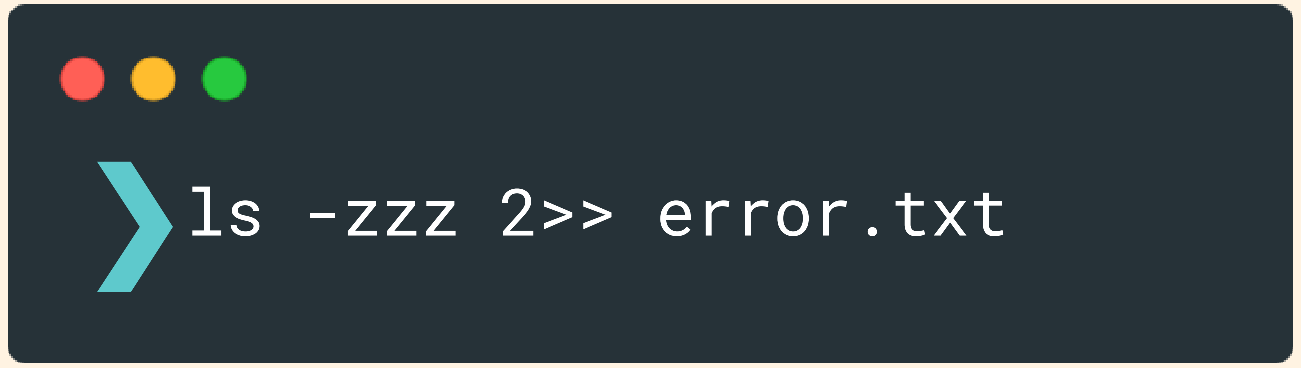


```
ls -zzz 2> error.txt
```

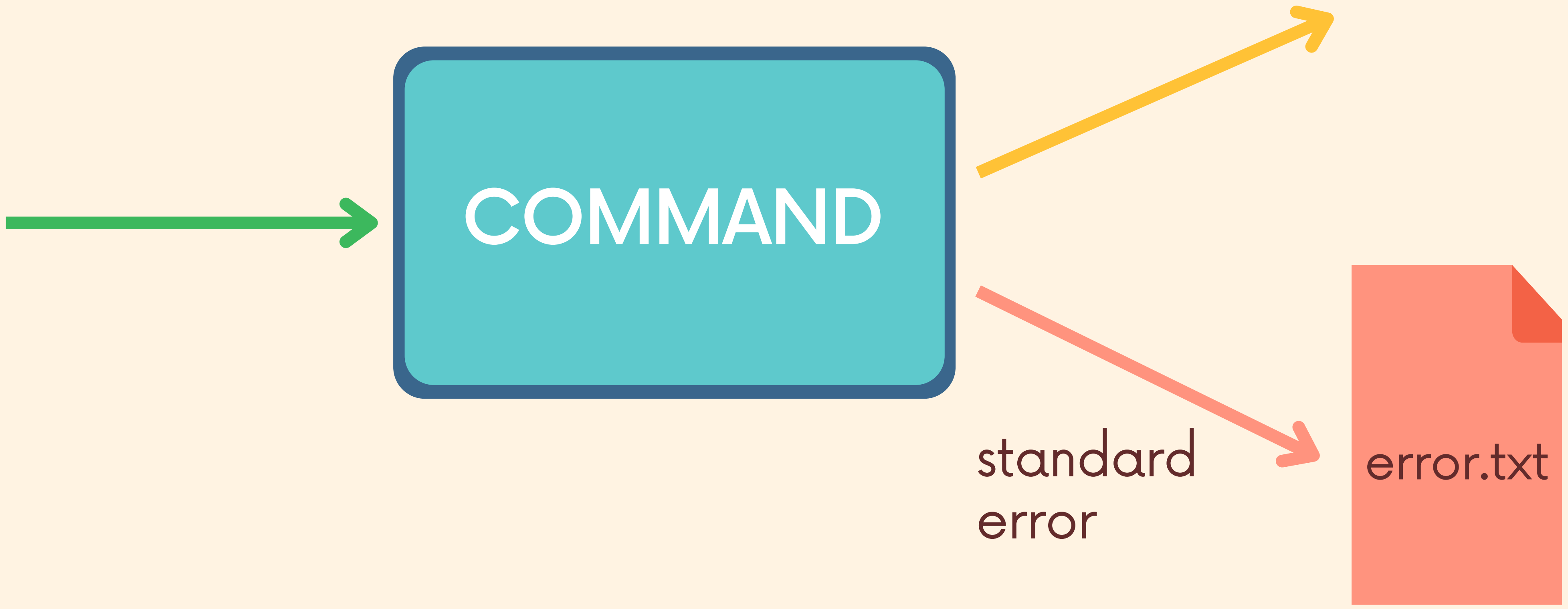


appending

We can use the same >> syntax to append when redirecting standard error.

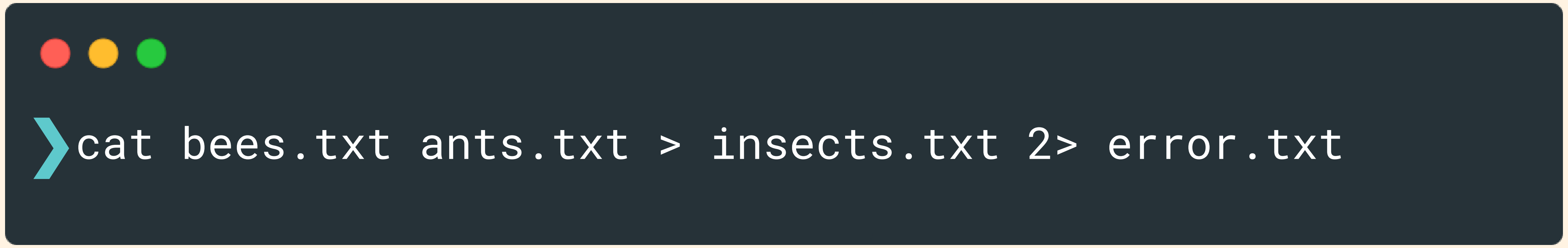


```
ls -zzz 2>> error.txt
```

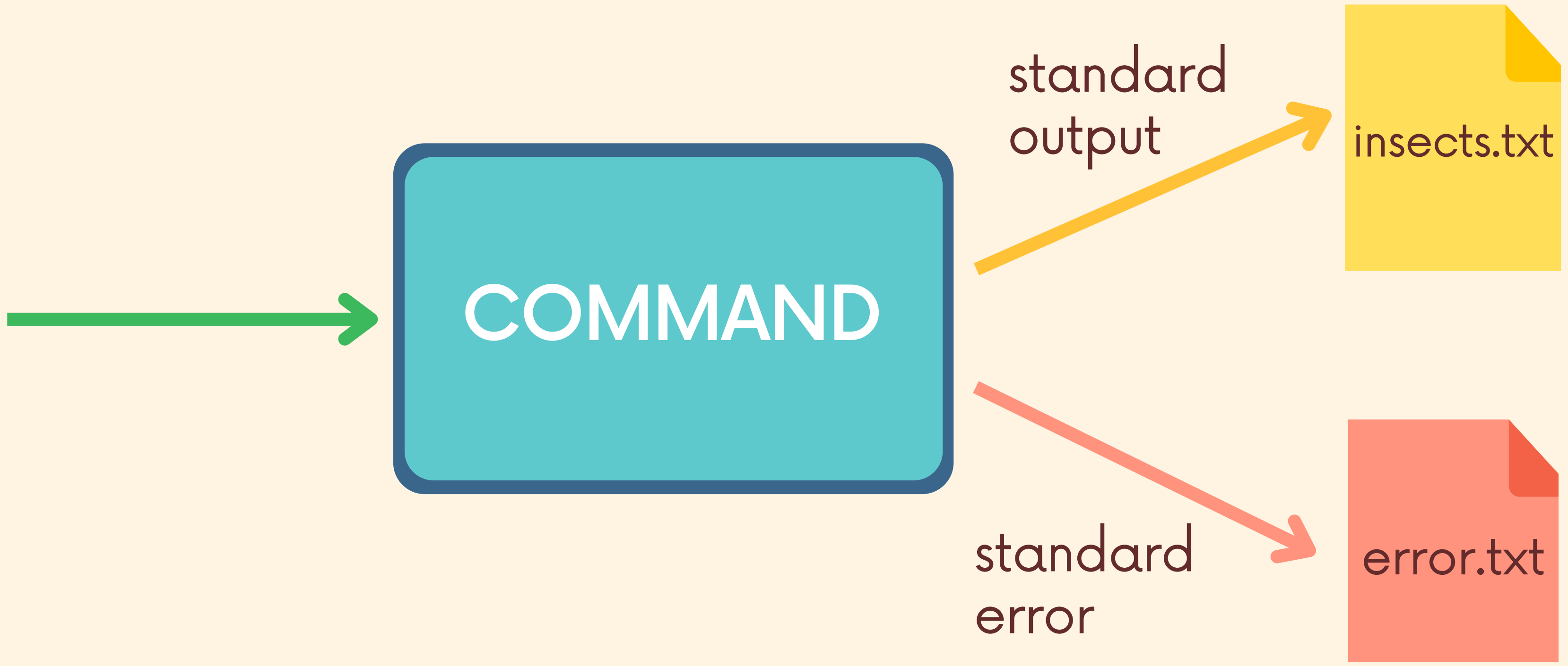


all together now

We can redirect multiple streams at once! In this example, we are concatenating two files, redirecting standard output to a file called insects.txt, and redirecting standard error to a file called error.txt

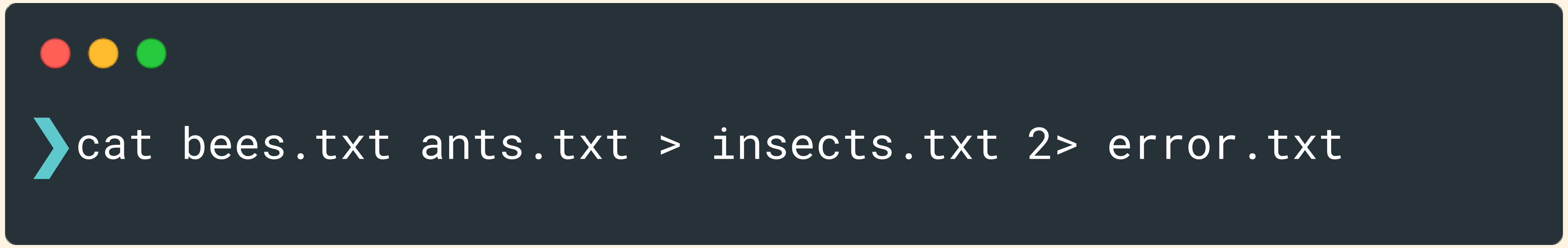


```
❯cat bees.txt ants.txt > insects.txt 2> error.txt
```

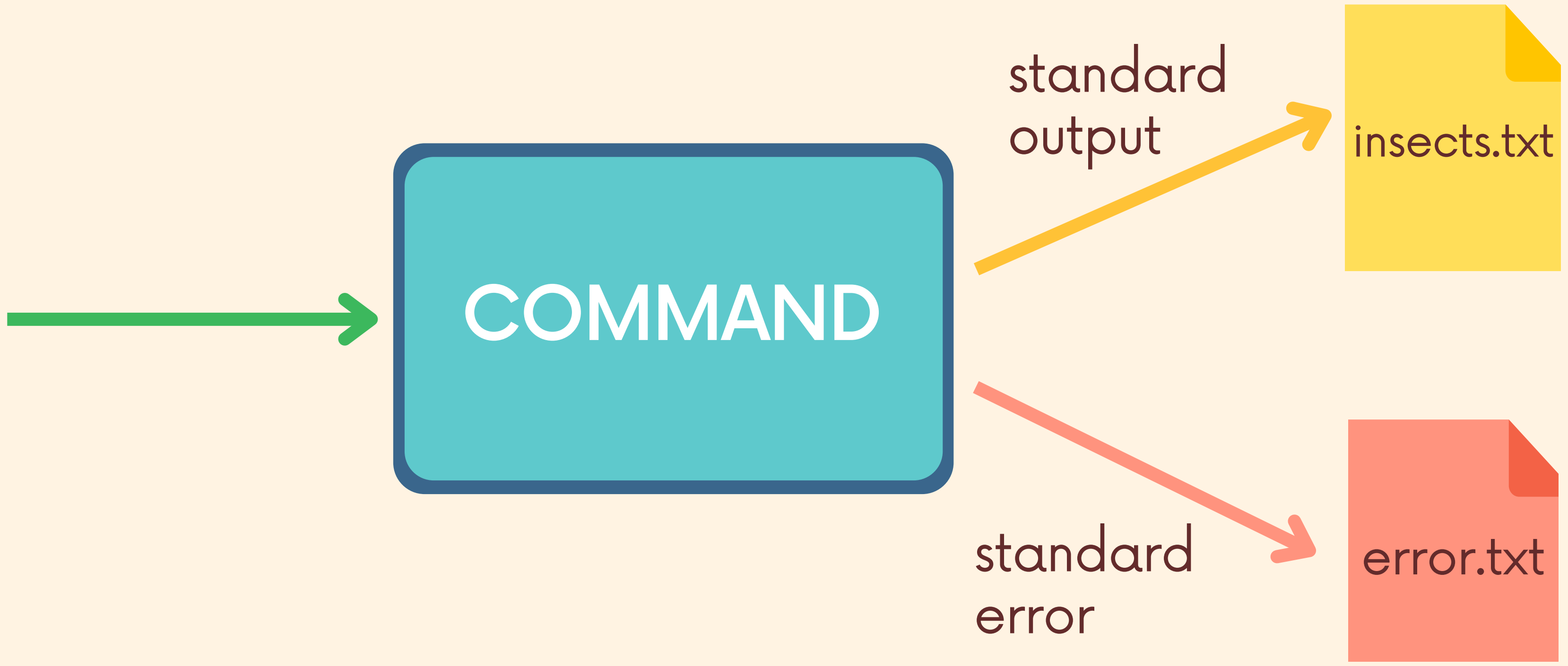


Order matters!

When redirecting both standard output and standard error, make sure standard output comes FIRST. Always redirect standard error after standard output.

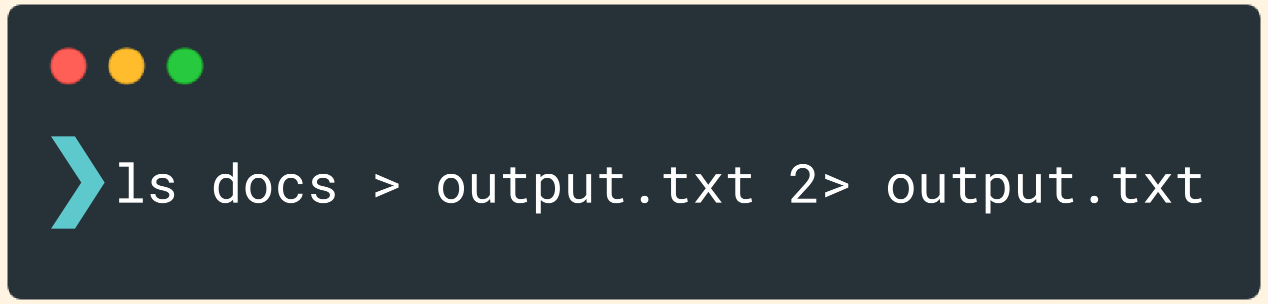


```
❯cat bees.txt ants.txt > insects.txt 2> error.txt
```



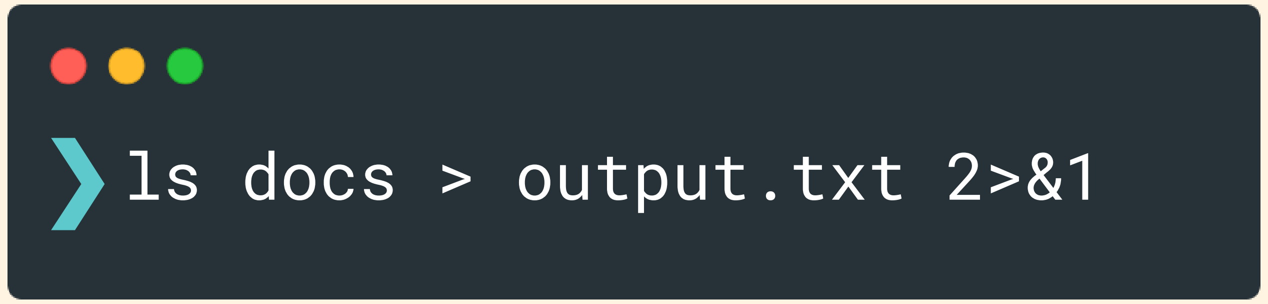
getting fancy

If we wanted to redirect both standard output and standard error to the same file, we could do this...

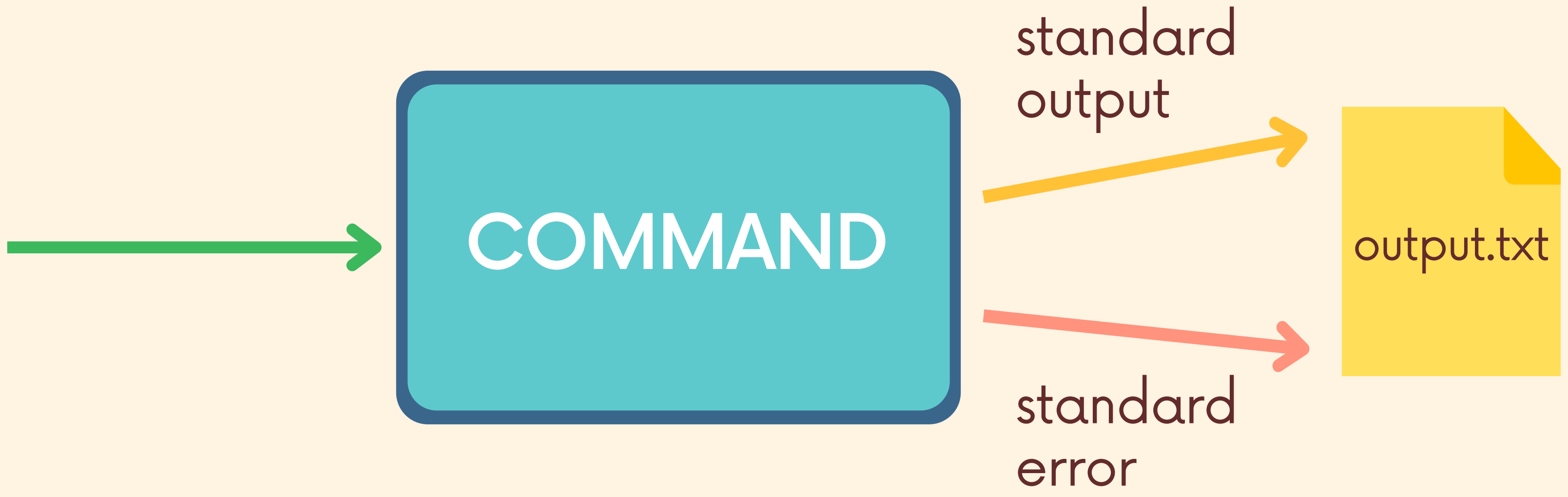


```
❯ls docs > output.txt 2> output.txt
```

Or we could instead use 2>&1 which is a fancy syntax for saying "redirect standard error to the same location as standard output (or whatever has the file descriptor #1)

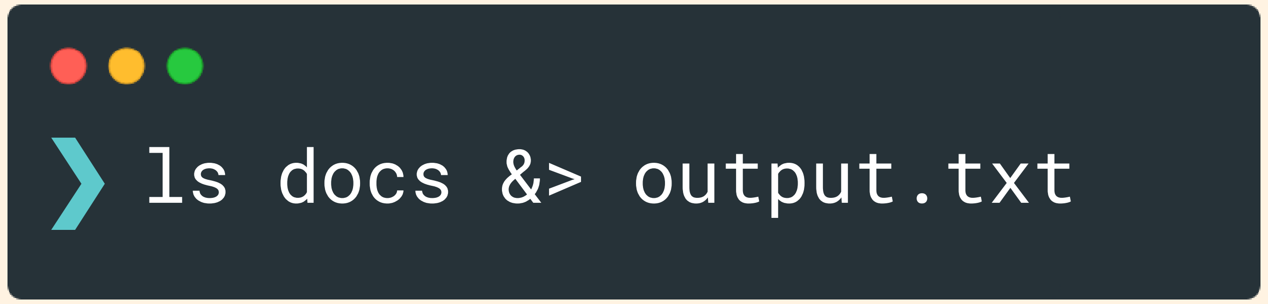


```
❯ ls docs > output.txt 2>&1
```

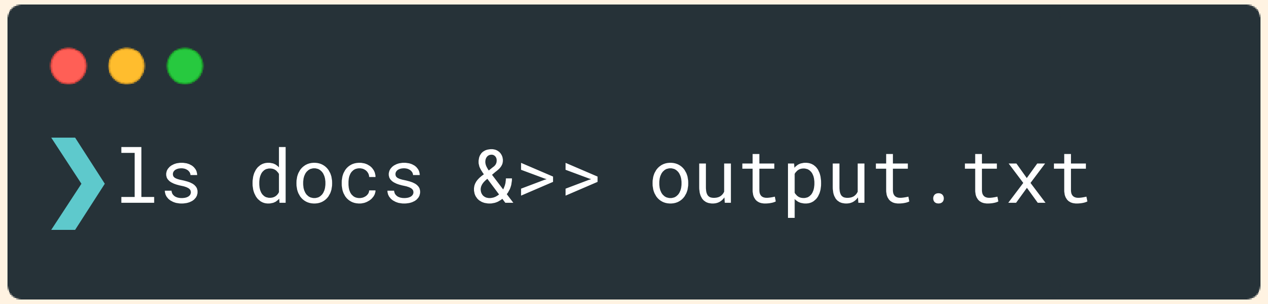


getting fancier

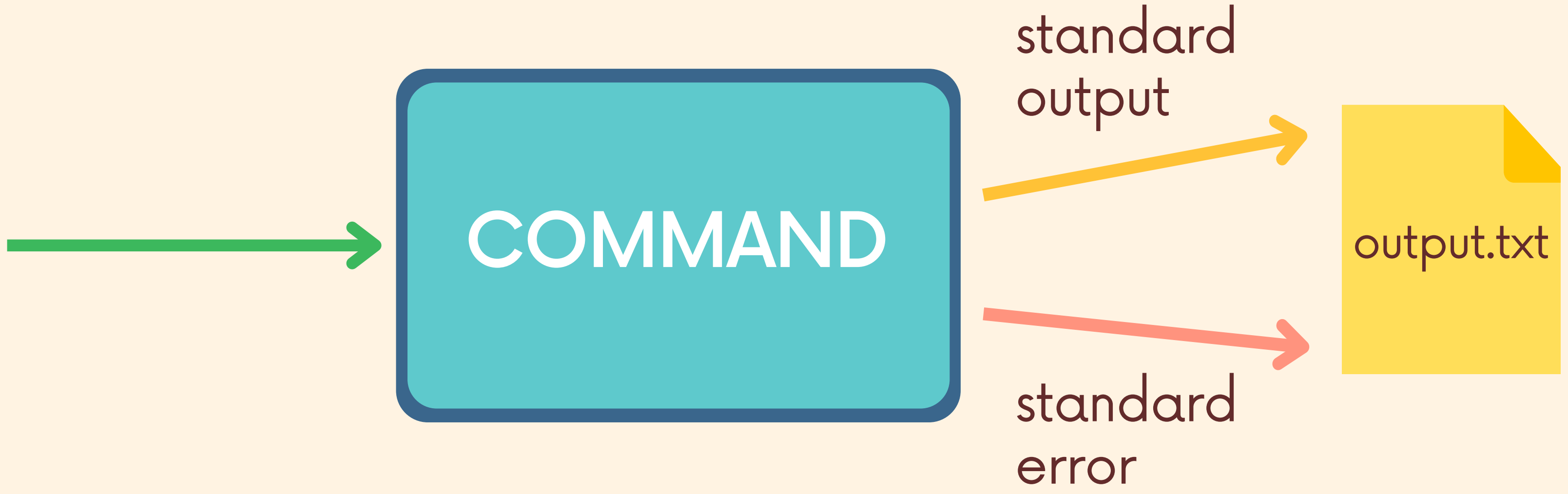
Newer versions of bash also support a fancier syntax for redirecting both standard output and standard error to the same file: the **&>** notation



```
❯ ls docs &> output.txt
```



```
❯ls docs &>> output.txt
```


standard
output
COMMAND
output.txt
standard
error