COMP 110/L Lecture 27

Kyle Dewey

Outline

- Reading from files

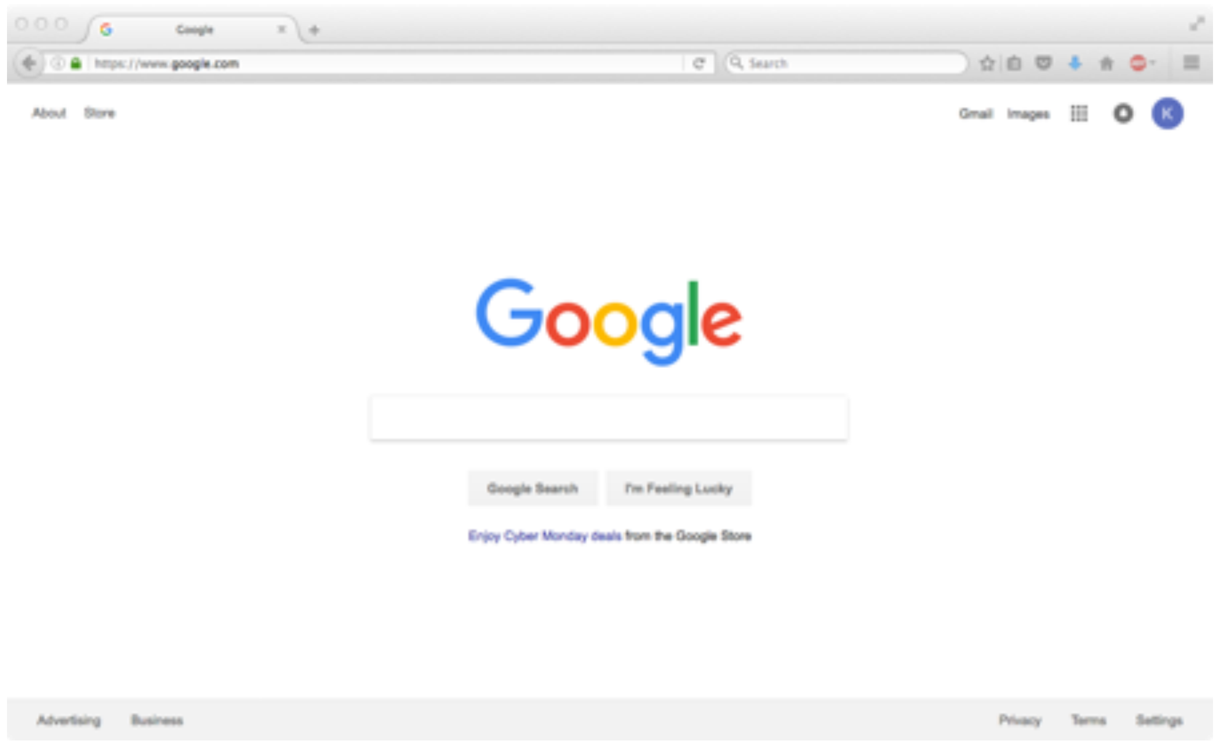
Reading From Files

Motivation

Files act like very large inputs; basis for most things.

Motivation

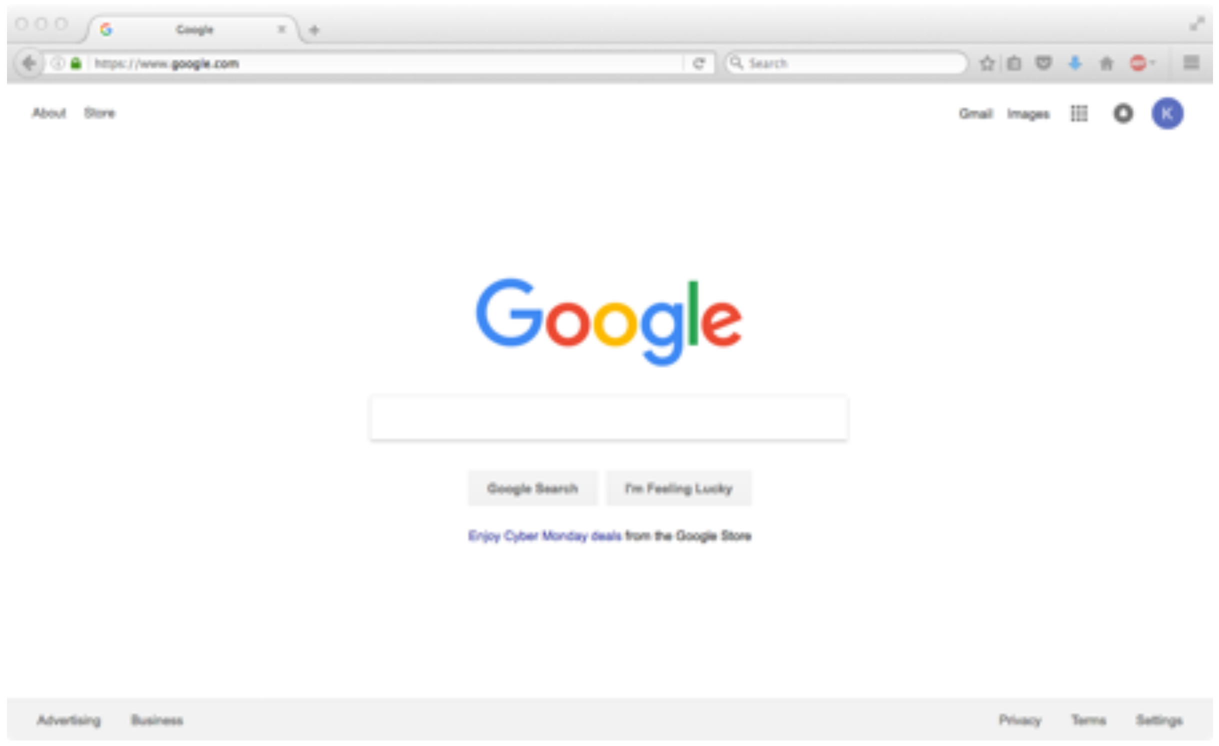
Files act like very large inputs; basis for most things.



–When you "access" a web page, you're really downloading a HTML file, and subsequently reading the file

Motivation

Files act like very large inputs; basis for most things.



```
public class MyClass {
    ...
}
```

–When you write code, the Java compiler will read it from the file.

Reading from Files

Reading from Files

```
myFile.txt
```

```
myFile.txt
```
Contents

```
one
two
three
```

-On disk somewhere, I have the file myFile.txt

Reading from Files

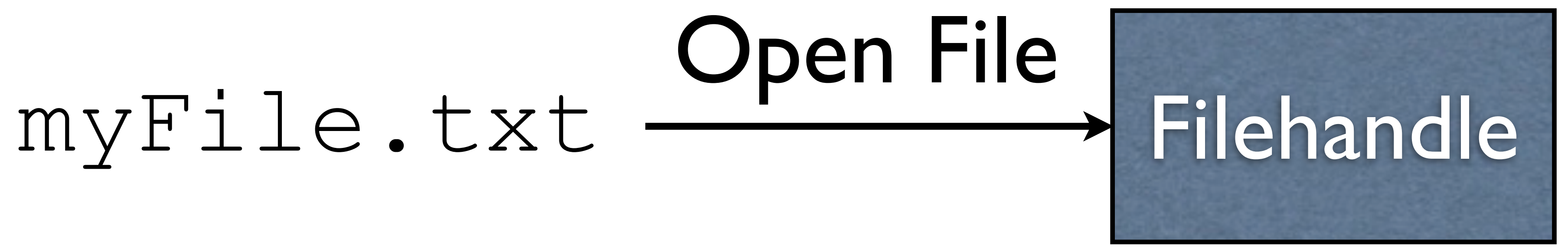
`myFile.txt` $\xrightarrow{\text{Open File}}$

`myFile.txt`
Contents

```
one
two
three
```

-On disk somewhere, I have the file myFile.txt

Reading from Files

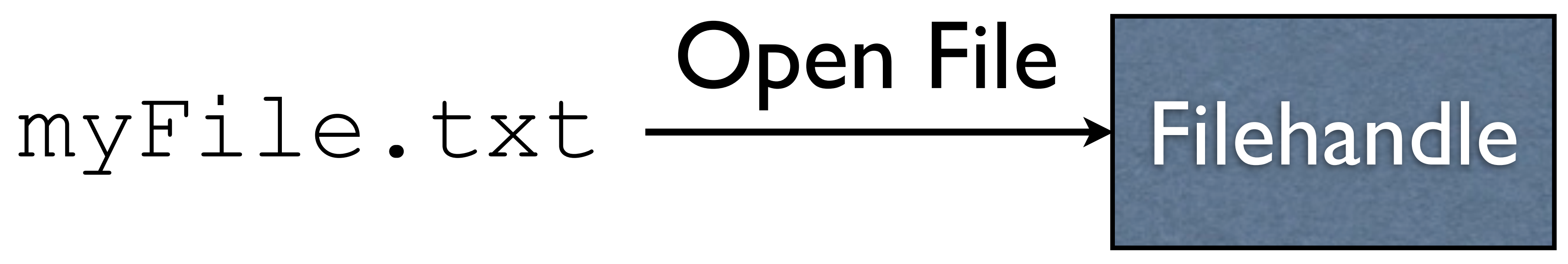


myFile.txt
Contents

```
one
two
three
```

-Opening a file creates a “filehandle”, that is, a handle on the open file.
-We call it a “handle” in much the same way as a pan has a handle – this is how to hold the pan (file) and manipulate the pan (file)

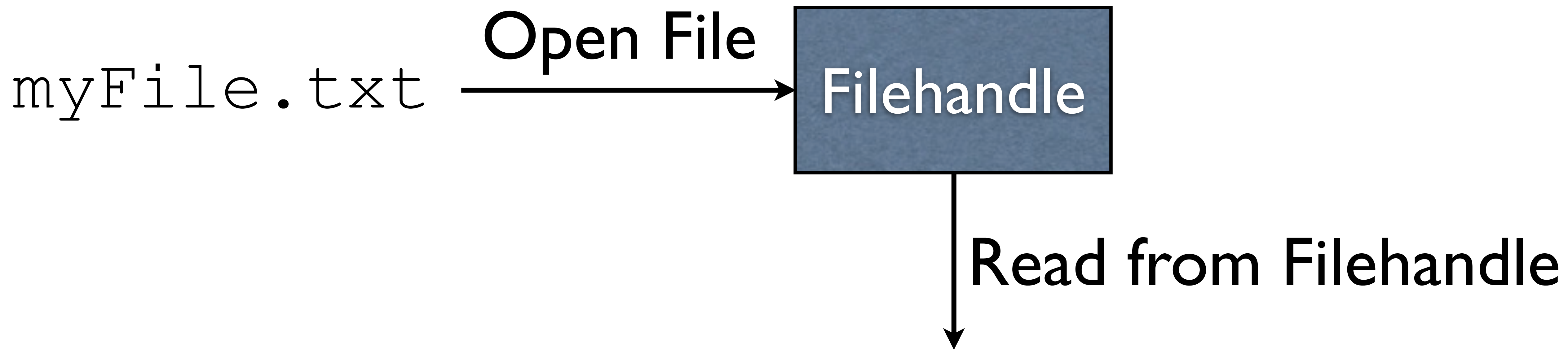
Reading from Files



myFile.txt
Contents
one
two
three

-The filehandle keeps track of where we are in the file
-Initially, we are right at the start of the file

Reading from Files

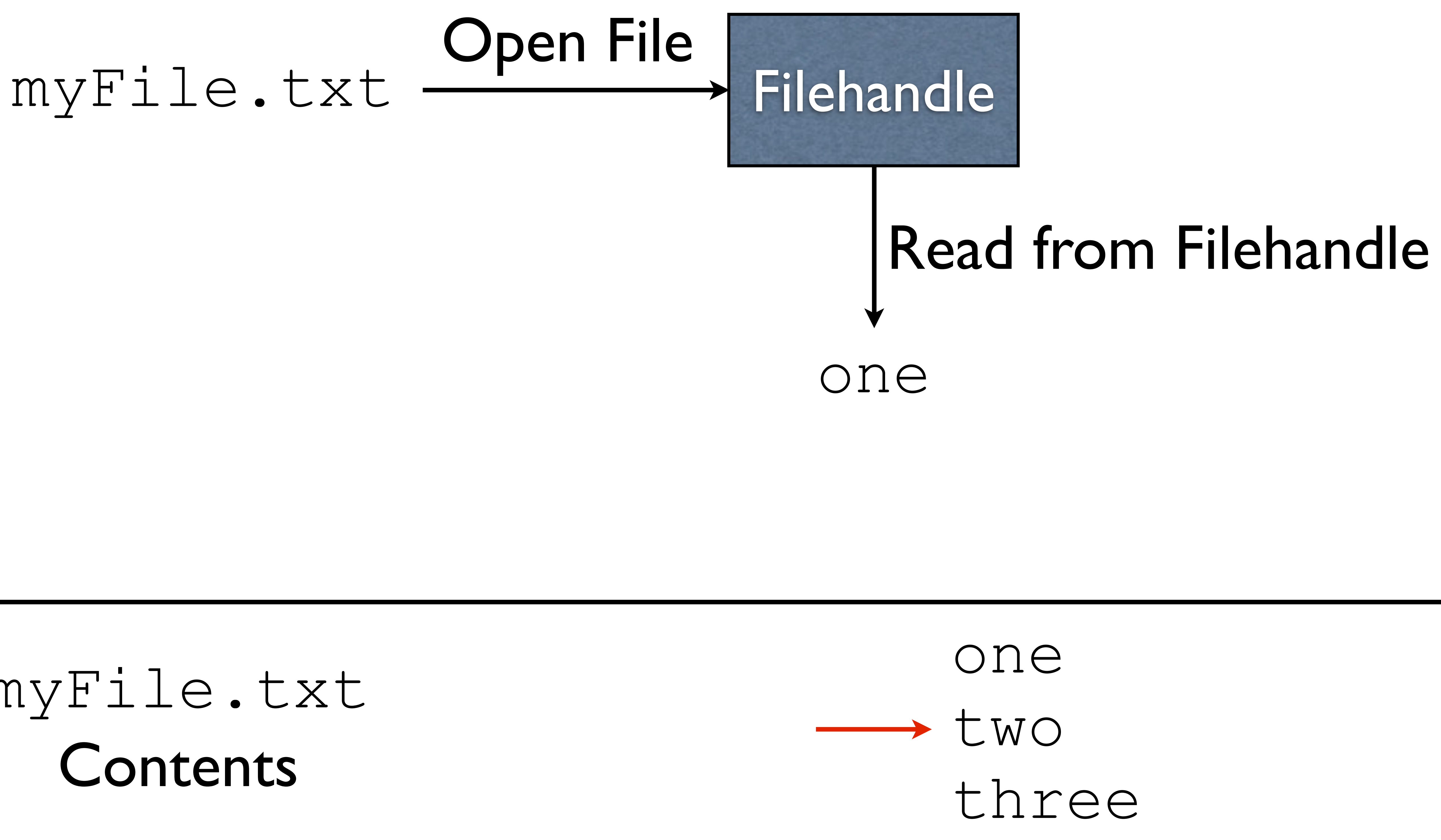


myFile.txt
Contents

```
one
two
three
```

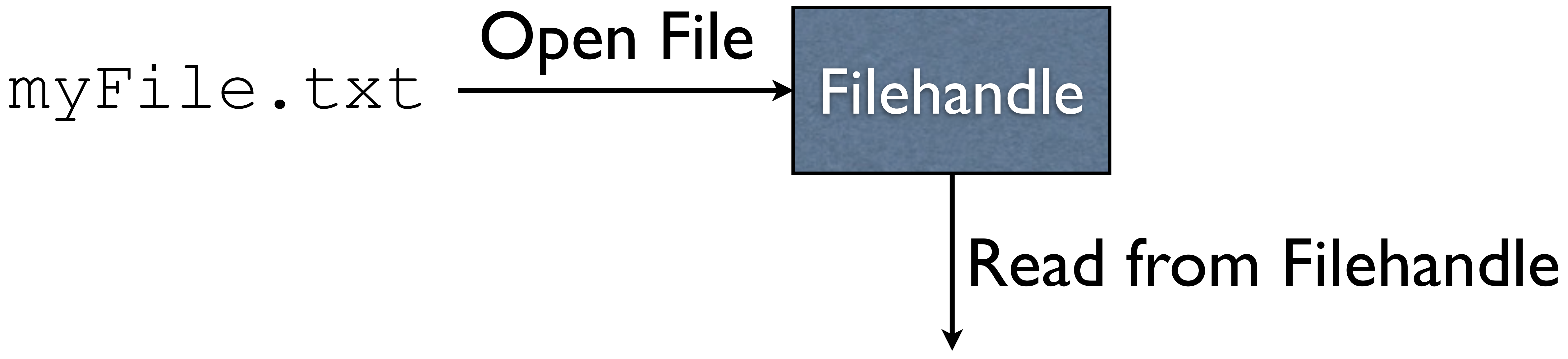
-We can then read from the filehandle

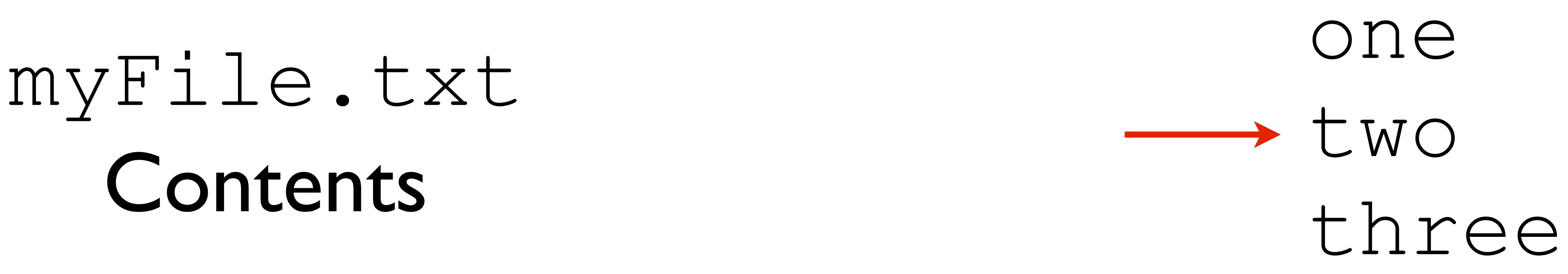
Reading from Files



-When we read from a filehandle, we get whatever is where the file pointer (the red arrow) is
-The file pointer is updated to point to the next position in the file

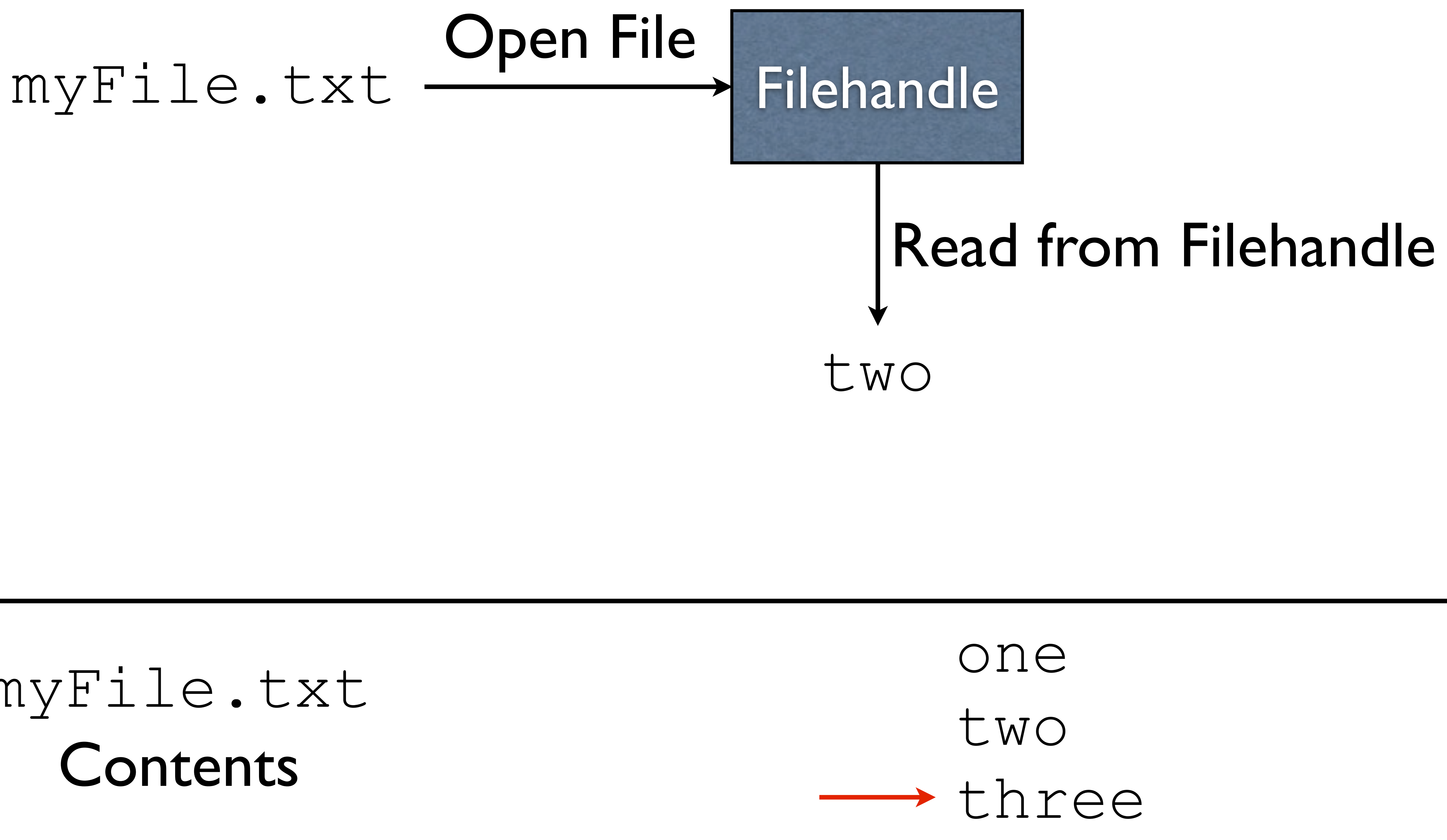
Reading from Files





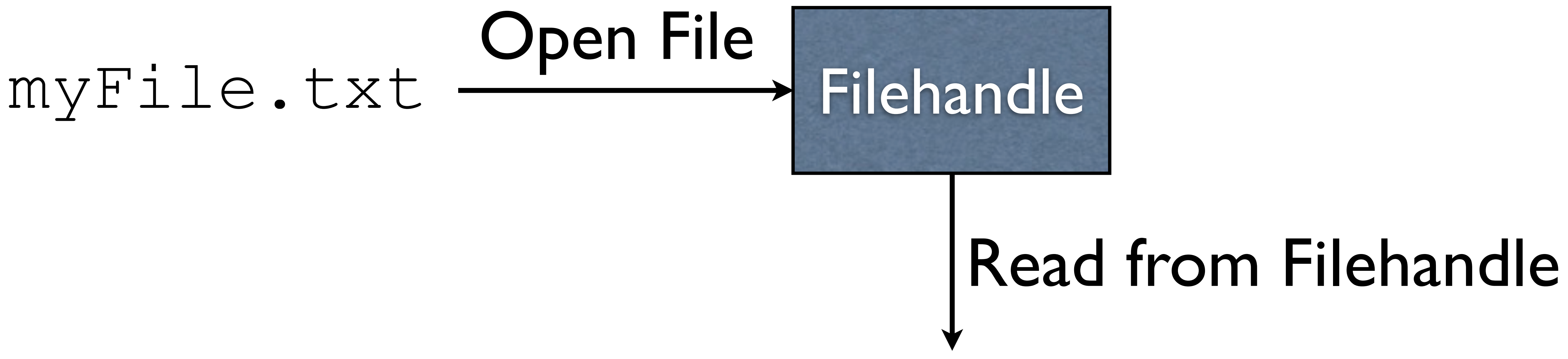
-We can then read again...

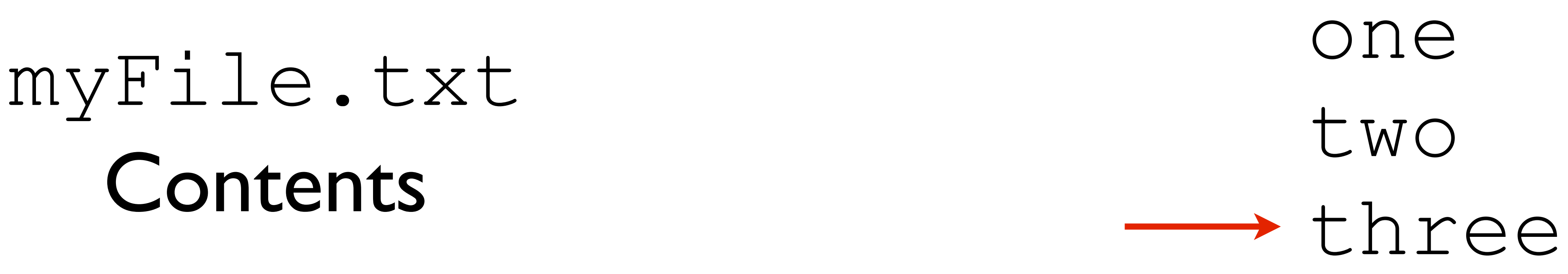
Reading from Files



-...resulting in the next value read from the file
-The file pointer (red arrow) is updated as before

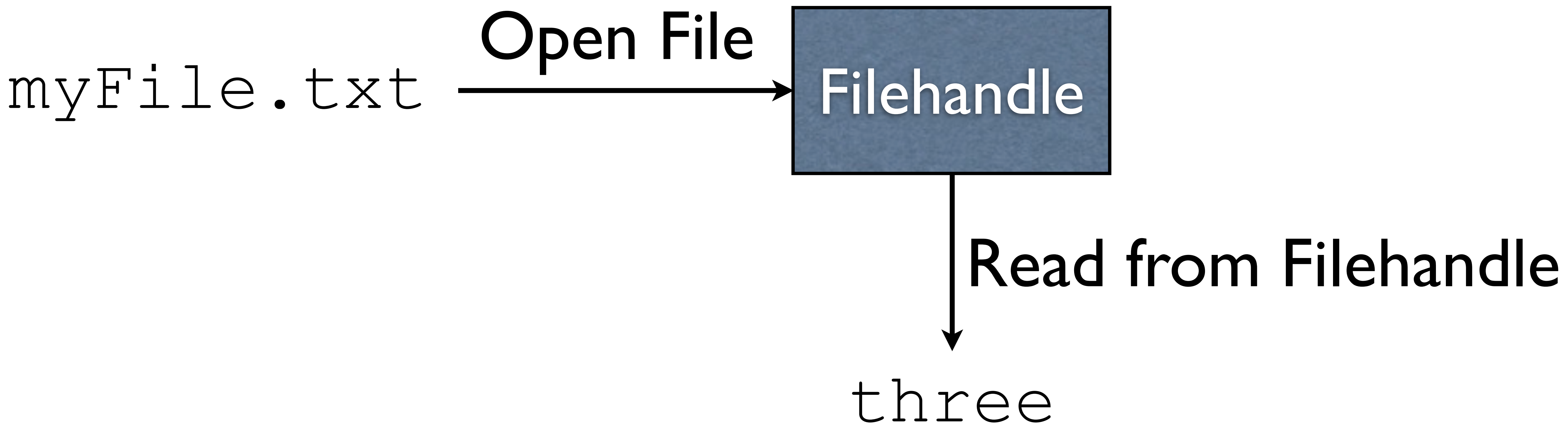
Reading from Files





-We can read again...

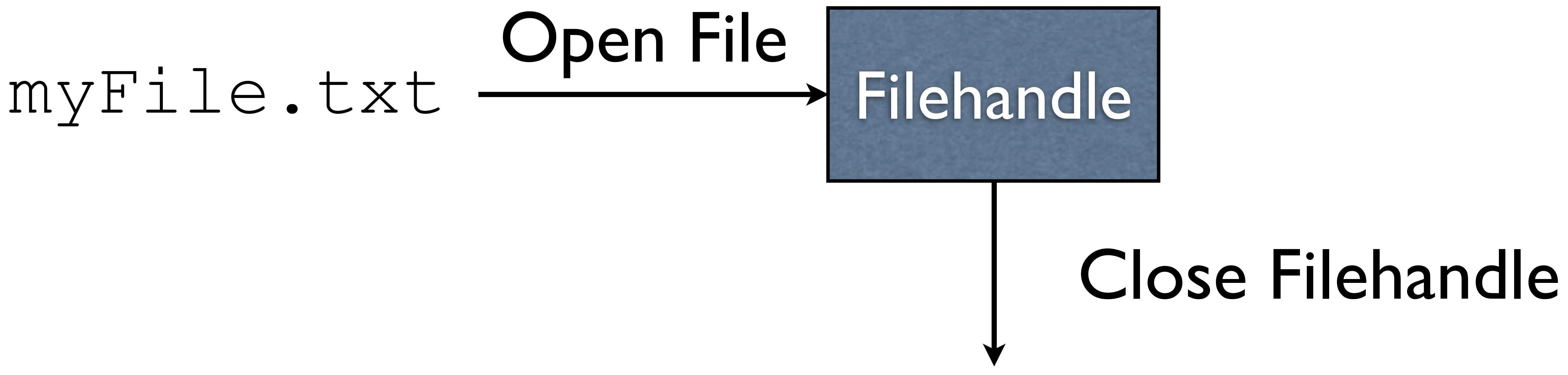
Reading from Files



myFile.txt
Contents
one
two
three

-...and we get the next thing with a file pointer update, as before

Reading from Files

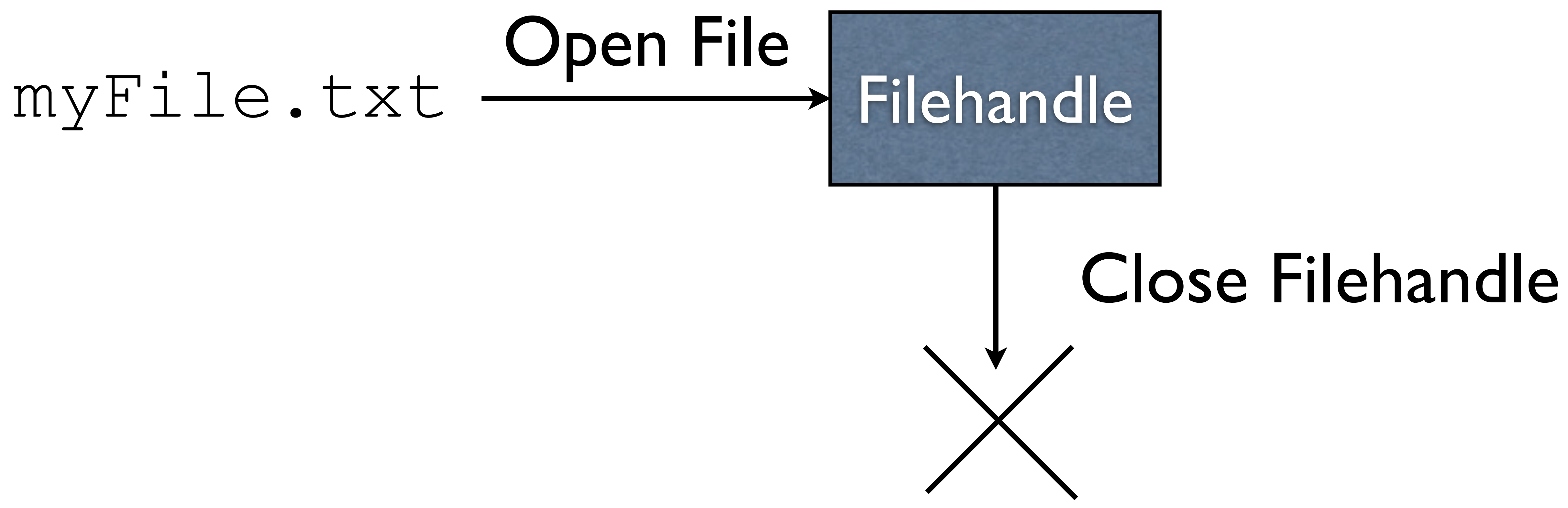


myFile.txt
Contents

```
one
two
three
```

-The last thing we do is close the filehandle when we are done with it

Reading from Files



myFile.txt
Contents

```
one
two
three
```

-Closing the filehandle doesn't visibly _do_ anything
-Internally, the file is no longer opened, and we no longer keep track of where we were in the file
-The underlying operating system puts a limit on how many files we can have open at once, so it's important to close a file when we're done with it.

Reading from Files with `Scanner`

Reading from Files with `Scanner`

Step 1: Create `File` object

Reading from Files with `Scanner`

Step 1: Create `File` object

```
File myFile = new File(“myFile.txt”);
```

Reading from Files with `Scanner`

Step 1: Create `File` object

```
File myFile = new File(“myFile.txt”);
```

Step 2: Create `Scanner` object with the `File` object

Reading from Files with `Scanner`

Step 1: Create `File` object

```
File myFile = new File("myFile.txt");
```

Step 2: Create `Scanner` object with the `File` object

```
Scanner input = new Scanner(myFile);
```

Reading from Files with `Scanner`

Step 1: Create `File` object

```
File myFile = new File(“myFile.txt”);
```

Step 2: Create `Scanner` object with the `File` object

```
Scanner input = new Scanner(myFile);
```

Step 3: Read from `Scanner` object

Reading from Files with `Scanner`

Step 1: Create `File` object

```
File myFile = new File("myFile.txt");
```

Step 2: Create `Scanner` object with the `File` object

```
Scanner input = new Scanner(myFile);
```

Step 3: Read from `Scanner` object

```
if (input.hasNextLine()) {
   String line = input.nextLine();
    ...
```

Reading from Files with `Scanner`

Step 4: Close `Scanner` object

Reading from Files with `Scanner`

Step 4: Close `Scanner` object

```
input.close();
```

Example:
`ReadFirstLine.java`

Example:
`ReadWholeFile.java`

`FileNotFoundException`

`Scanner`'s constructor will throw a `FileNotFoundException` if the file does not exist.

`FileNotFoundException`

`Scanner`'s constructor will throw a `FileNotFoundException` if the file does not exist.

Example:
`ReadWholeFileWithTry.java`