## Writing to a File

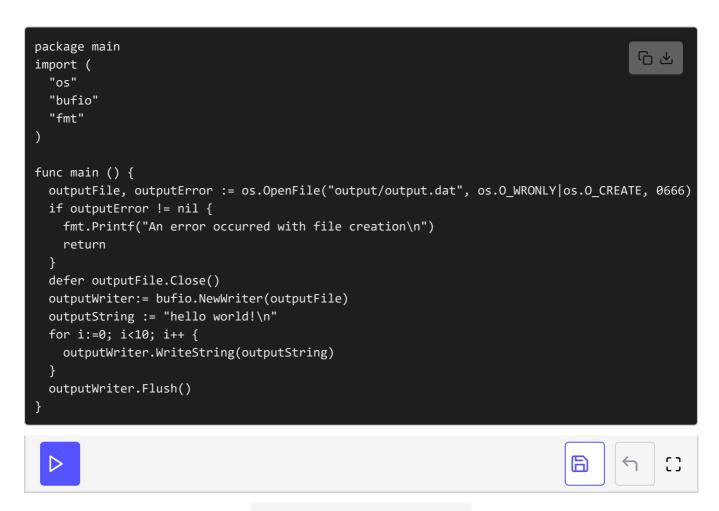
This lesson provides code examples and their explanations for writing to a file.

we'll cover the following ^

Writing data to a file

## Writing data to a file #

Writing data to a file is demonstrated in the following program:



Writing using bufio Package

Apart from a filehandle, we now need a writer from bufio. We open a file output.dat for write-only at line 9. The file is created if it does not exist.

The as Open Tile is a function that allows more control over the eneming mode

of a file. We see that the OpenFile function takes a filename, one or more flags

(logically OR-d together using the | bitwise OR operator if more than one) and the file permissions to use. The following flags are commonly used:

- os.O\_RDONLY: the read flag for read-only access
- os.O\_WRONLY: the write flag for write-only access
- os.O\_RDWR: the flag that provides both read and write access
- os.O\_CREATE: the create flag to create the file if it doesn't exist
- os.O\_TRUNC: the truncate flag to truncate to size 0 if the file already exists

When reading, the file permissions are ignored so we can use a value of 0. When writing, we use the standard Unix file permissions of 0666 (even on Windows).

The error-handling is done from **line 10** to **line 13**, and at **line 14**, we assure the file will be closed with the **defer** keyword.

Then at **line 15**, we make the writer-object (the buffer) called **outputWriter**. We make a simple **hello world!\n** string, which will be written *ten* times to the buffer in the for-loop (see **line 18**). The buffer is then written completely to the file at **line 20**. In simple write tasks, this can be done more efficiently with:

```
fmt.Fprintf(outputFile, "Some test data.\n")
```

Using the F version of the fmt print functions that can write to any io.Writer, including a file.

The program below illustrates an alternative to fmt.Fprintf:

```
package main
import "os"

func main() {
  os.Stdout.WriteString("hello, world\n")
  f, _ := os.OpenFile("output/test.txt", os.O_CREATE|os.O_WRONLY, 0)
  defer f.Close()

  f.WriteString("hello, world in a file\n")
}
```

## Writing using io Package

In the code above, at **line 5**, instead of printing with <code>fmt</code>, we use the <code>Stdout.WriteString</code> method from <code>os</code> to display a string. At **line 6**, we open a file **test.txt** for writing. At **line 9**, we are writing a string to it. Because of the <code>defer</code> at <code>line 7</code>, the file will be closed at <code>line 10</code>.

With os.Stdout.WriteString("hello, world\n"), we can also write to the screen. In f, \_ := os.OpenFile("test", os.O\_CREATE|os.O\_WRONLY, 0) we create or open for write-only a file **test.txt**. A possible error is disregarded with \_. We don't make use of a buffer, we write immediately to the file with: f.WriteString().

Now that you are familiar with the write operations, the next lesson brings you a challenge to solve.