Урок Спринт 1.2

Чтение файла

Чтение и запись файлов — базовые операции для многих приложений. Сегодня вы узнаете, как работать с файлами в Go. Но для начала определимся с терминами.

Файл — это набор данных (например, программа, текст, фото), который сохранён на носителе информации. Подробнее о файлах можно прочесть **тут**.

Чтение файла

Допустим, у нас есть файл `literature.txt` со списком литературы на лето. Давайте прочитаем его:

```
data, err := os.ReadFile("literature.txt")
```

Теперь в объекте `data` есть содержимое нашего файла. Отлично! Урок можно заканчивать!

Но что, если файл очень большой — например, ваш любимый фильм в отличном качестве? Когда мы попытаемся его прочесть, всё содержимое файла скопируется в объект `data`.

Почему это плохо? Во-первых, такой дубль файла займёт всё свободное место в оперативной памяти. Во-вторых, программа будет читать его очень долго, а мы хотим, чтобы она работала быстрее.

Как исправить положение? Мы можем просто открыть файл, но не читать его:

```
f, err := os.Open("movie.mkv")
```

В таком случае в объекте `f` будет не содержимое файла, а его дескриптор, с помощью которого мы прочтём сам файл:

```
buffer := make([]byte, 100) // будем считывать файл в этот слайс
_, err := f.Read(buffer) // читаем часть файла
```

`buffer` — это объект, куда мы будем читать файл. После чтения в нём находится 100 байтов нашего

файла — либо ещё меньше, если сам файл небольшой.

Как же нам прочитать весь файл с фильмом? Мы будем делать это последовательно:

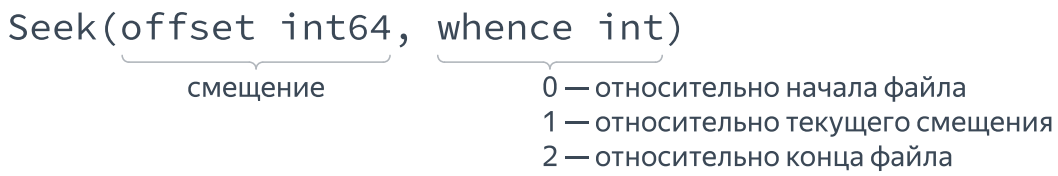
```
buffer := make([]byte, 100)
for {
    // n — число байтов, которые удалось прочитать
    n, err := f.Read(buffer) // прочитаем часть в buffer
    if err == io.EOF { // ошибка показывает, что файл прочитан полностью
        break // выходим из цикла, файл прочитан
    }
    if err != nil { // эту ошибку нужно обработать
        // ошибка чтения
    }
    process(buffer[:n]) // функция для дальнейшей обработки
}
```

Вернёмся к `literature.txt`. Предположим, в нём хранятся данные по всей школьной программе литературы. Для того, чтобы прочесть этот файл, лучше также воспользоваться дескриптором. Считывать определённое количество байтов в буфер будет неудобно — программа может обрезать строку, если она не поместится в нужный объём.

Было бы гораздо удобнее считывать файл построчно. Вот как это сделать:

```
f, err := os.Open("movie.mkv")
fileScanner := bufio.NewScanner(f) //NewScanner возвращает специальный сканер для чтения построч
for fileScanner.Scan() { // сканер перемещается к следующему токену, то есть к следующей части т
    process(fileScanner.Text()) // функция для обработки строки с литературой — например, здесь
}
```

И ещё раз про большие файлы. Предположим, у вас есть текстовый файл с логами работы программы. Вам нужно прочесть только последние записи из него. Самый простой способ — прочесть файл от начала до конца, но оптимальнее будет применить функцию Seek. С её помощью можно читать файл не с начала, а с выбранного места: Вот как можно её использовать:




```
offset := 1024
f.Seek(offset, 0) // сместимся от начала файла
buffer := make([]byte, 100)
n, err := f.Read(buffer) // прочитаем в buffer с позиции 1024
```

Справка