07

# Lendo múltiplas linhas

### Transcrição

Se queremos que o nosso programa leia além da primeira linha do arquivo, o código responsável por fazer a leitura deve ser executado mais de uma vez. Então, o que vamos fazer é colocar esse código dentro de um `for`, até dar um erro específico, o erro de `EOF` (*End Of File*), que acontece quando não há mais linha a serem lidas.

```
// restante do código omitido

func leSitesDoArquivo() []string {

    var sites []string

    arquivo, err := os.Open("sites.txt")
    if err != nil {
        fmt.Println("Ocorreu um erro:", err)
    }

    leitor := bufio.NewReader(arquivo)

    for {
        linha, err := leitor.ReadString('\n')
        fmt.Println(linha)
        if err == io.EOF {
            fmt.Println("Ocorreu um erro:", err)
        }
    }

    return sites
}
```

Mas como saímos do `for`? Se houver o erro, que já estamos verificando, nós damos um `break`, que faz com que o código saia do loop:

```
// restante do código omitido

func leSitesDoArquivo() []string {

    var sites []string

    arquivo, err := os.Open("sites.txt")
    if err != nil {
        fmt.Println("Ocorreu um erro:", err)
    }

    leitor := bufio.NewReader(arquivo)

    for {
        linha, err == leitor.ReadString('\n')
        fmt.Println(linha)
        if err == io.EOF {
```

```
            break
        }
    }

    return sites
}
```

Ao iniciar o monitoramento, vemos a seguinte impressão:

```
https://random-status-code.herokuapp.com/

https://www.alura.com.br

https://www.caelum.com.br
```

A impressão está sendo desse jeito pois estamos pulando linha no arquivo, então ela é lida também. O leitor lê a linha, incluindo o `\n`, por isso que fica esse pulo de linha na hora de impressão.

Logo, devemos remover essa quebra de linha da linha lida, antes de adicioná-la ao slice. Para isso, existe a função `TrimSpace`, do pacotes `strings`, que remove as quebras de linha e espaços ao final de uma string:

```
// restante do código omitido

func leSitesDoArquivo() []string {

    var sites []string

    arquivo, err := os.Open("sites.txt")
    if err != nil {
        fmt.Println("Ocorreu um erro:", err)
    }

    leitor := bufio.NewReader(arquivo)

    for {
        linha, err := leitor.ReadString('\n')
        linha = strings.TrimSpace(linha)
        fmt.Println(linha)
        if err == io.EOF {
            break
        }
    }

    return sites
}
```

Agora, as linhas são impressas sem quebra de linha. Logo, já podemos adicioná-las ao slice:

```
// restante do código omitido

func leSitesDoArquivo() []string {
```

```
    var sites []string

    arquivo, err := os.Open("sites.txt")
    if err != nil {
        fmt.Println("Ocorreu um erro:", err)
    }

    leitor := bufio.NewReader(arquivo)

    for {
        linha, err := leitor.ReadString('\n')
        linha = strings.TrimSpace(linha)
        sites = append(sites, linha)
        if err == io.EOF {
            break
        }
    }

    return sites
}
```

Ao executar o programa novamente e iniciar o monitoramento, podemos perceber que os sites são monitorados corretamente! Agora, se quisermos adicionar mais sites, basta colocá-los no arquivo **sites.txt**.

Por último, devemos ser educados com o sistema operacional, se abrimos um arquivo com `os.Open` , após lê-lo, é uma boa prática fechá-lo com a função `Close` :

```
// restante do código omitido

func leSitesDoArquivo() []string {

    var sites []string

    arquivo, err := os.Open("sites.txt")
    if err != nil {
        fmt.Println("Ocorreu um erro:", err)
    }

    leitor := bufio.NewReader(arquivo)

    for {
        linha, err := leitor.ReadString('\n')
        linha = strings.TrimSpace(linha)
        sites = append(sites, linha)
        if err == io.EOF {
            break
        }
    }

    arquivo.Close()

    return sites
}
```