

# Kata Find and Replace

## Énoncé

Programme qui trouve et remplace un mot par un autre dans un fichier.

## Exemple

Remplacer le mot **Go** par **Python**

Source: wikigo.txt	Résultat
Go was conceived in 2007 to improve programming productivity at Google	Python was conceived in 2007 to improve programming productivity at Pythonogle

## Features

- Affiche le résumé du traitement en console
  - Nombre d'occurrences du mot
  - Numéros de lignes modifiés
- Écrire le texte modifié dans un nouveau fichier pour préserver l'original
- Bonus : prendre aussi en compte que le mot peut avoir une majuscule (ou pas !)

## Exemple de résumé

```
$ go run main.go
== Summary ==
Number of occurrences of Go: 10
Number of lines: 7
Lines: [ 1 - 8 - 15 - 17 - 19 - 23 - 28 ]
== End of Summary ==
```

## Prototypes

```
func ProcessLine(line, oldWord, newWord string) (found bool, result string, occurrences int)
```

- `line` : ligne à traiter
- `oldWord` / `newWord` : ancien mot / nouveau mot
- `found` : vrai si au moins une occurrence trouvée
- `result` : résultat après traitement
- `occurrences` : nombre d'occurrences de `oldWord` dans la ligne

```
func FindReplaceFile(src string, dst string, oldWord string, newWord string) (occurrences int, lines []int, err error)
```

- `src` : nom du fichier source
- `oldWord` / `newWord` : ancien mot / nouveau mot
- `occurrences` : nombre d'occurrences de `oldWord`
- `lines` : tableau des numéros de lignes où `oldWord` a été trouvé
- `err` : erreur générée par la fonction

### Astuce

`FindReplaceFile` n'arrive pas à ouvrir le fichier, il faut renvoyer une erreur

# Outils

## Bufio

```
scanner := bufio.NewScanner(srcFile)
for scanner.Scan() {
    t := scanner.Text()
}
```

```
writer := bufio.NewWriter(dstFile)
defer writer.Flush()
fmt.Fprintln(writer, "Texte d'une ligne")
```

## Strings

```
c := strings.Contains("go ruby java", "go") // c == true
cnt := strings.Count("go go go", "go") // cnt == 3
res := strings.Replace("old go", "go", "python", -1) // res == "old python"
```

# Solution

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func ProcessLine(line, oldWord, newWord string) (found bool, result string, occurrences int) {
    oldWordLower := strings.ToLower(oldWord)
    newWordLower := strings.ToLower(newWord)
    result = line
    if strings.Contains(line, oldWord) || strings.Contains(line, oldWordLower) {
        found = true
        occurrences += strings.Count(line, oldWord)
        occurrences += strings.Count(line, oldWordLower)
        result = strings.ReplaceAll(line, oldWord, newWord)
        result = strings.ReplaceAll(result, oldWordLower, newWordLower)
    }

    return found, result, occurrences
}

func FindReplaceFile(src string, dst string, oldWord string, newWord string) (occurrences int, lines []int, err error) {
    // open src file
    srcFile, errSrc := os.Open(src)
    if errSrc != nil {
        return 0, lines, errSrc
    }
    defer srcFile.Close()

    // open dst file
    dstFile, errDst := os.Create(dst)
    if errDst != nil {
        return 0, nil, errDst
    }
```

```

    }
    defer dstFile.Close()

    oldWord = oldWord
    newWord = newWord
    lineIdx := 1
    scanner := bufio.NewScanner(srcFile)
    writer := bufio.NewWriter(dstFile)
    defer writer.Flush()

    for scanner.Scan() {
        found, res, o := ProcessLine(scanner.Text(), oldWord, newWord)
        if found {
            occurrences += o
            lines = append(lines, lineIdx)
        }

        _, errWrt := fmt.Fprintln(writer, res)
        if errWrt != nil {
            return occurrences, lines, errWrt
        }

        lineIdx++
    }

    return occurrences, lines, nil
}

func main() {
    oldWord := "Go"
    newWord := "Python"
    occurrences, lines, err := FindReplaceFile("wikigo.txt", "wikipython.txt", oldWord, newWord)
    if err != nil {
        fmt.Printf("Error while executing find replace: %v\n", err)
        return
    }

    fmt.Println("== Summary ==")
    defer fmt.Println("== End of Summary ==")
    fmt.Printf("Number of occurrences of %s: %d\n", oldWord, occurrences)
}

```

```
fmt.Printf("Number of lines: %d\nLines: [ ", len(lines))
linesCount := len(lines)
for i, l := range lines {
    fmt.Printf("%d", l)
    if i < linesCount-1 {
        fmt.Printf(" - ")
    }
}
fmt.Println(" ]")

}
```

Revision #1

Created 9 May 2022 19:15:20 by Noé Larrieu-Lacoste

Updated 9 May 2022 19:17:42 by Noé Larrieu-Lacoste