# LIRE ET ÉCRIRE DANS UN FICHIER DANS LE LANGAGE DE PROGRAMMATION GO

## Introduction

Dans ce chapitre, nous allons voir comment vous pouvez efficacement lire et écrire sur des fichiers à l'aide du langage de programmation go.

Il existe de multiples façons pour lire et écrire dans un fichier. Dans les exemples qui suivent je vais vous montrer les techniques que j'utilise selon les différents cas d'utilisation que je juge le plus simple.

Je vais manipuler un fichier nommé **test.txt** avec le contenu suivant :

```
je suis un fichier de test
```

## Lire seulement un fichier

Dans le cas ou vous ne faites que lire un fichier, le mieux reste d'utiliser la fonction `ReadFile()` de la bibliothèque `io/ioutil`. Cette fonction retourne un type `byte` (bit), il faut donc penser à **caster** (convertir le type) le résultat obtenu en `string`

```
package main

import (
    "fmt"
    "io/ioutil"
)

func main() {
    data, err := ioutil.ReadFile("test.txt") // lire le fichier text.txt
    if err != nil {
        fmt.Println(err)
```

```
    }

    fmt.Println(string(data)) // conversion de byte en string
}
```

**Résultat :**

```
je suis un fichier de test
```

Dans le cas ou le fichier n'existe pas, vous aurez l'erreur suivante :

```
Le fichier spécifié est introuvable.
```

# Écrire sur un fichier

il y a deux manières pour écrire dans un fichier. Soit vous décidez d'**écraser** un fichier après **écriture**, soit d'écrire à la suite du contenu du fichier.

Pour écrire dans un fichier on va utiliser la **bibliothèque** `os`.

```
package main

import (
    "fmt"
    "io/ioutil"
    "os"
)

func main() {
    file, err := os.OpenFile("test.txt", os.O_CREATE|os.O_WRONLY|os.O_APPEND, 0600)
    defer file.Close() // on ferme automatiquement à la fin de notre programme

    if err != nil {
        panic(err)
    }

    _, err = file.WriteString("test\n") // écrire dans le fichier
    if err != nil {
        panic(err)
    }

    _, err = file.WriteString("i love test\n")
```

```
    if err != nil {
        panic(err)
    }

    data, err := ioutil.ReadFile("test.txt") // lire le fichier
    if err != nil {
        fmt.Println(err)
    }

    fmt.Print(string(data))
}
```

> **Information**
>
> J'utilise le "\n" pour faire un saut de la ligne dans mon fichier.

**Résultat :**

```
je suis un fichier de test
test
i love test
```

Je reviens ici sur ces lignes de code :

```
file, err := os.OpenFile("test.txt", os.O_CREATE|os.O_WRONLY|os.O_APPEND, 0600)
defer file.Close() // on ferme automatiquement le fichier après l'avoir manipulé
```

La fonction `os.OpenFile()`, propose vraiment beaucoup d'options, que je vous explique ci-dessous :

- Premier paramètre : correspond au nom du fichier à ouvrir
- Deuxième paramètre : ici ce sont des options spécialement dédiées au fichier que vous allez manipuler :
    - `os.O_CREATE` : Permet de créer le fichier si il n'existe pas.

    - `os.O_WRONLY` : Permet de rendre le fichier (dans votre programme) accessible en écriture seulement.
    - `os.O_APPEND` : Permet de ne pas écraser le fichier quand vous écrivez dessus (supprimez cette option si vous souhaitez écraser le fichier).
- Troisième paramètre : les droits d'accès de votre fichier (Plus d'information sur les permissions ici ici)

**Il est absolument important de fermer le fichier à la fin de votre programme**. D'où l'utilisation de la fonction `close()`, j'ai rajouté le mot clé `defer`, ce mot-clé permet d'exécuter la ligne de code en question jusqu'à la fin d'exécution d'une fonction.

## Bonus

On se retrouve avec beaucoup de lignes de code répétables, il serait temps d'utiliser les super pouvoirs des fonctions pour mieux structurer notre code :

```
package main

import (
    "fmt"
    "io/ioutil"
    "os"
)

func check(e error) {
    if e != nil {
        panic(e)
    }
}

func write(text string, file *os.File) {
    if _, err := file.WriteString(text); err != nil {
        panic(err)
    }
}

func read(filename string) string {
    data, err := ioutil.ReadFile(filename)
```

```
    check(err)
    return string(data)
}

func main() {
    file, err := os.OpenFile("test.txt", os.O_CREATE|os.O_WRONLY|os.O_APPEND, 0600)
    defer file.Close()
    check(err)

    write("Test\n", file)

    data := read(file.Name())
    fmt.Print(data)
}
```

**Résultat :**

```
je suis un fichier de test
test
i love test
Test
```

# Exercice.

Voilà maintenant vous savez comment lire et écrire dans un fichier. Si vous souhaitez aller encore plus loin, vous pouvez reprendre le tp du morpions pour rajouter un système qui vous permet de sauvegarder la partie en cours (tour du joueur et le damier).