Fichiers

Pour manipuler un fichier en Go, il existe plusieurs librairies permettant différentes actions.

io/ioutil

C'est sans doute l'approche la plus simple pour manipuler un fichier.

Elle permet de directement lire un répertoire ou le contenu d'un fichier, et même d'écrire dedans.

Lecture fichier

```
func readFile(filename string) (error) {
	dat, err:= ioutil.ReadFile(filename)
	if err != nil {
		return "", err
	}

	if len(dat) == 0 {
		// return "", errors.New("Empty content")
		return "", fmt.Errorf("Empty content (filename=%v)", filename)
	}
	
	fmt.Printf("%s\n", dat)

	return nil
}
```

Écriture fichier

```
func writeFile(filename, content string) error {
	err:= ioutil.WriteFile(filename, []byte(content), 0644)
	if err != nil {
		return err
```

```
	}
	return nil
}
```

Lecture répertoire

```
func readDir() error {
	files, err := ioutil.ReadDir(".")
	if err != nil {
		return err
	}

	for _, file := range files {
		fmt.Println(file.Name())
	}
	return nil
}
```

Inconvénient

On fait de la lecture / écriture direct, aucun buffer.

Peut poser problème en cas de traitement de gros fichiers....

On peut potentiellement écraser un fichier existant

os + bufio

`Bufio` implémente des manipulations de flux avec des buffers, un peu plus verbeux, mais beaucoup plus intéressant !

Lecture

```
func readFile(filename string) error {
	srcFile, errSrc := os.Open(src)
	if errSrc != nil {
```

```
    return errSrc
  }

  lineIdx := 1
  scanner := bufio.NewScanner(srcFile)

  for ; scanner.Scan(); lineIdx++ {
    fmt.Println("Line", lineIdx, ":", scanner.Text())
  }

  srcFile.Close()
  return nil
}
```

Écriture

```
func writeFile(filename string, lines []string) error {
  dstFile, errDst := os.Create(dst)
  if errDst != nil {
    return errDst
  }

  writer := bufio.NewWriter(dstFile)

  for _, line range lines {
    _, errWrt := fmt.Fprintln(writer, line)
    if errWrt != nil {
      return errWrt
    }
  }

  writer.Flush()
  dstFile.Close()
  return nil
}
```

Revision #1
Created 2022-05-09 19:13:31 UTC by Noé Larrieu-Lacoste
Updated 2022-05-09 19:13:54 UTC by Noé Larrieu-Lacoste