Présentation du module Directory

Jean-Luc JOULIN

Version 1
18 septembre 2022

Sommaire

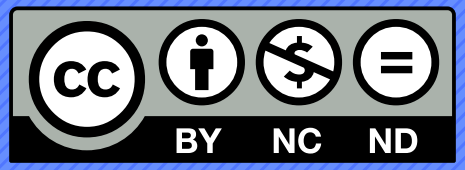
License

Cette présentation est diffusée suivant les termes de la license Creative Common :

BY **Attribution**. Cette présentation peut être librement utilisée, à condition de l'attribuer à l'auteur en citant son nom.

NC **Pas d'utilisation Commerciale**. Aucune utilisation commerciale n'est permise.

ND **Pas de Modification**. Aucune œuvre dérivée basée sur cette présentation n'est autorisée.



Module Directory

Présentation du module

- Module permettant d'effectuer des opérations sur les fichiers et les répertoires.
- Nom complet System.Directory.
- Nécessite d'être chargé dans un fichier source avec import System.Directory.
- Nécessite d'ajouter une dépendance dans le fichier .cabal.

```
build-depends:
  directory >= 1.3,
```

- Description complète du module sur le site Hackage.

Tests

Existence

doesFileExist :: FilePath → IO Bool

Test l'existence d'un fichier.

```
ghci> doesFileExist "/home/jean-luc/.bashrc"
True
ghci> doesFileExist "/home/jean-luc/other.txt"
False
```

doesDirectoryExist :: FilePath → IO Bool

Test l'existence d'un répertoire.

```
ghci> doesDirectoryExist "/home/jean-luc/"
True
ghci> doesDirectoryExist "/home/jean-luc/rep"
False
```

Actions sur les répertoires

Création de répertoires

createDirectory :: FilePath → IO ()

Créé un nouveau répertoire.

```
Prelude> createDirectory "mydir"
```

createDirectoryIfMissing :: Bool → FilePath → IO ()

Créé un nouveau répertoire si il n'éxiste pas. Si le premier argument est True, les répertories parents sont aussi créés.

```
Prelude> createDirectoryIfMissing True "mydir"
```

Contenu des répertoires

listDirectory :: FilePath → IO [FilePath]

Retourne une liste contenant les éléments d'un répertoire.

```
Prelude> listDirectory "~"
*** Exception: ~: getDirectoryContents: does not exist (No such file
    or directory)
Prelude> listDirectory "/home/jean-luc/Programmation/Haskell/tests/"
["Listing_Exemple_01.txt","tab2fu.txt","test_list.txt",".back"]
```

getDirectoryContents :: FilePath → IO ()

Retourne une liste contenant les éléments d'un répertoire en incluant les entrées spéciales "." et "..".

```
Prelude> getDirectoryContents "
    /home/jean-luc/Programmation/Haskell/tests/"
["Listing_Exemple_01.txt","tab2fu.txt","test_list.txt",".back",".","..
    "]
```

Suppression de répertoires

removeDirectory :: FilePath → IO ()

Supprime un répertoire.

```
Prelude> removeDirectory "mydir"
```

removeDirectoryRecursive :: FilePath → IO()

Supprime un répertoire et tout son contenu.

```
Prelude> removeDirectoryRecursive "mydir"
```

Répertoires de travail

getCurrentDirectory :: IO FilePath

Retourne le répertoire de travail.

```
ghci> getCurrentDirectory
"/home/jean-luc/Programmation/Haskell/tests"
```

setCurrentDirectory :: FilePath → IO ()

Définie le répertoire de travail.

```
ghci> setCurrentDirectory "/home/jean-luc/"
ghci> getCurrentDirectory
"/home/jean-luc"
```

Répertoires spécifiques

getHomeDirectory :: IO FilePath

Retourne le répertoire de travail.

```
ghci> getHomeDirectory
"/home/jean-luc"
```

getTemporaryDirectory :: IO FilePath

Retourne le répertoire temporaire.

```
ghci> getTemporaryDirectory
"/tmp"
```

Actions sur les fichiers

Renommage et suppression de fichiers

removeFile :: FilePath → IO ()

Retourne le répertoire de travail.

```
ghci> removeFile "myfiletodelete"
```

renameFile :: FilePath → FilePath → IO ()

Renomme un fichier.

```
ghci> renameFile "filename" "newfilename"
```

Copie de fichiers

```
copyFile :: FilePath → FilePath → IO ()
```

Copie un fichier vers une destination.

```
ghci> copyFile "myfile" "/home/jean-luc/destination/"
```

```
copyFileWithMetadata :: FilePath → FilePath → IO ()
```

Copie un fichier vers une destination en préservant les permissions et la date de modification.

```
ghci> copyFileWithMetadata "myfile" "/home/jean-luc/destination/"
```

Actions sur les permissions

Obtention des permissions

getPermissions :: FilePath → IO Permissions

Obtenir les permissions d'un fichier.

```
ghci> getPermissions "myfile"
Permissions {readable = True, writable = True, executable = False,
    searchable = False}
```

readable :: Permissions → Bool

Test l'attribut lecture de permissions.

```
ghci> perm <- getPermissions "myfile"
ghci> readable perm
True
```

writable :: Permissions → Bool

Test l'attribut écriture de permissions.

```
ghci> perm <- getPermissions "myfile"
ghci> writable perm
True
```

Application des permissions

setPermissions :: FilePath → Permissions → IO ()

Applique les permissions sur un fichier.

```
ghci> setPermissions "myfile.txt" (setOwnerWritable False
    emptyPermissions)
ghci> setPermissions "myfile.txt" (setOwnerExecutable True
    emptyPermissions)
```

copyPermissions :: FilePath → FilePath → IO ()

Copie les permissions d'un fichier vers un autre fichier.

```
ghci> copyPermissions "basefile" "destinationfile"
```

Actions sur les dates

Récupération des dates

getAccessTime :: FilePath → IO UTCTime

Retourne la date d'accès

```
ghci> getAccessTime "tab3fu.txt"
2019-04-02 13:10:44.507556305 UTC
```

getModificationTime :: FilePath → IO UTCTime

Retourne la date de modification

```
ghci> getModificationTime "tab3fu.txt"
2019-04-02 13:10:42.011596057 UTC
```

Coordonnées

Coordonnées

Jean-Luc JOULIN
jean-luc-joulin@orange.fr
www.jeanjoux.fr