JFP **13** (1): 219–222, January 2003. © 2003 Cambridge University Press
DOI: 10.1017/S0956796803002417 Printed in the United Kingdom

Chapter 22

Directory Functions

```
module Directory (
    Permissions( Permissions, readable, writable, executable, searchable ),
    createDirectory, removeDirectory, removeFile,
    renameDirectory, renameFile, getDirectoryContents,
    getCurrentDirectory, setCurrentDirectory,
    doesFileExist, doesDirectoryExist,
    getPermissions, setPermissions,
    getModificationTime ) where

import Time ( ClockTime )

data Permissions = Permissions {
                       readable,   writable,
                       executable, searchable :: Bool
                   }

instance Eq   Permissions where ...
instance Ord  Permissions where ...
instance Read Permissions where ...
instance Show Permissions where ...

createDirectory         :: FilePath -> IO ()
removeDirectory         :: FilePath -> IO ()
removeFile              :: FilePath -> IO ()
renameDirectory         :: FilePath -> FilePath -> IO ()
renameFile              :: FilePath -> FilePath -> IO ()
```


https://doi.org/10.1017/S0956796803002417 Published online by Cambridge University Press

```
getDirectoryContents    :: FilePath -> IO [FilePath]
getCurrentDirectory     :: IO FilePath
setCurrentDirectory     :: FilePath -> IO ()

doesFileExist           :: FilePath -> IO Bool
doesDirectoryExist      :: FilePath -> IO Bool

getPermissions          :: FilePath -> IO Permissions
setPermissions          :: FilePath -> Permissions -> IO ()

getModificationTime     :: FilePath -> IO ClockTime
```

These functions operate on directories in the file system.

Any `Directory` operation could raise an `isIllegalOperation`, as described in Section 21.1; all other permissible errors are described below. Note that, in particular, if an implementation does not support an operation it should raise an `isIllegalOperation`. A directory contains a series of entries, each of which is a named reference to a file system object (file, directory etc.). Some entries may be hidden, inaccessible, or have some administrative function (for instance, "." or ".." under POSIX), but all such entries are considered to form part of the directory contents. Entries in sub-directories are not, however, considered to form part of the directory contents. Although there may be file system objects other than files and directories, this library does not distinguish between physical files and other non-directory objects. All such objects should therefore be treated as if they are files.

Each file system object is referenced by a *path*. There is normally at least one absolute path to each file system object. In some operating systems, it may also be possible to have paths which are relative to the current directory.

Computation `createDirectory` dir creates a new directory dir which is initially empty, or as near to empty as the operating system allows.

Error reporting. The `createDirectory` computation may fail with: `isPermissionError` if the user is not permitted to create the directory; `isAlreadyExistsError` if the directory already exists; or `isDoesNotExistError` if the new directory's parent does not exist.

Computation `removeDirectory` dir removes an existing directory dir. The implementation may specify additional constraints which must be satisfied before a directory can be removed (for instance, the directory has to be empty, or may not be in use by other processes). It is not legal for an implementation to partially remove a directory unless the entire directory is removed. A conformant implementation need not support directory removal in all situations (for instance, removal of the root directory).

Computation `removeFile` $file$ removes the directory entry for an existing file $file$, where $file$ is not itself a directory. The implementation may specify additional constraints which must be satisfied before a file can be removed (for instance, the file may not be in use by other processes).

https://doi.org/10.1017/S0956796803002417 Published online by Cambridge University Press

Error reporting. The `removeDirectory` and `removeFile` computations may fail with `isPermissionError` if the user is not permitted to remove the file/directory; or `isDoesNotExistError` if the file/directory does not exist.

Computation `renameDirectory` old new changes the name of an existing directory from old to new. If the new directory already exists, it is atomically replaced by the old directory. If the new directory is neither the old directory nor an alias of the old directory, it is removed as if by `removeDirectory`. A conformant implementation need not support renaming directories in all situations (for instance, renaming to an existing directory, or across different physical devices), but the constraints must be documented.

Computation `renameFile` old new changes the name of an existing file system object from old to new. If the new object already exists, it is atomically replaced by the old object. Neither path may refer to an existing directory. A conformant implementation need not support renaming files in all situations (for instance, renaming across different physical devices), but the constraints must be documented.

Error reporting. The `renameDirectory` and `renameFile` computations may fail with: `isPermissionError` if the user is not permitted to rename the file/directory, or if either argument to `renameFile` is a directory; or `isDoesNotExistError` if the file/directory does not exist.

Computation `getDirectoryContents` dir returns a list of *all* entries in dir. Each entry in the returned list is named relative to the directory dir, not as an absolute path.

If the operating system has a notion of current directories, `getCurrentDirectory` returns an absolute path to the current directory of the calling process.

Error reporting. The `getDirectoryContents` and `getCurrentDirectory` computations may fail with: `isPermissionError` if the user is not permitted to access the directory; or `isDoesNotExistError` if the directory does not exist.

If the operating system has a notion of current directories, `setCurrentDirectory` dir changes the current directory of the calling process to dir.

Error reporting. `setCurrentDirectory` may fail with: `isPermissionError` if the user is not permitted to change directory to that specified; or `isDoesNotExistError` if the directory does not exist.

The `Permissions` type is used to record whether certain operations are permissible on a file/ directory. `getPermissions` and `setPermissions` get and set these permissions, respectively. Permissions apply both to files and directories. For directories, the `executable` field will be `False`, and for files the `searchable` field will be `False`. Note that directories may be searchable without being readable, if permission has been given to use them as part of a path, but not to examine the directory contents.

Note that to change some, but not all permissions, a construct on the following lines must be used.

```
makeReadable f = do
                   p <- getPermissions f
                   setPermissions f (p {readable = True})
```

https://doi.org/10.1017/S0956796803002417 Published online by Cambridge University Press

The operation `doesDirectoryExist` returns `True` if the argument file exists and is a directory, and `False` otherwise. The operation `doesFileExist` returns `True` if the argument file exists and is not a directory, and `False` otherwise.

The `getModificationTime` operation returns the clock time at which the file/directory was last modified.

Error reporting. `get(set)Permissions`, `doesFile(Directory)Exist` and `getModificationTime` may fail with: `isPermissionError` if the user is not permitted to access the appropriate information; or `isDoesNotExistError` if the file/directory does not exist. The `setPermissions` computation may also fail with: `isPermissionError` if the user is not permitted to change the permission for the specified file or directory; or `isDoesNotExistError` if the file/directory does not exist.

https://doi.org/10.1017/S0956796803002417 Published online by Cambridge University Press