Directory Functions

The following functions can be used
to examine directory contents or manipulate directories.

paths.dir(dname)

Return a table containing the files and directories in directory `dname`.
This function return `nil` if the specified directory does not exists.
For linux, this includes the `.` and `..` directories.

paths.files(dname [, include])

Returns an iterator over the files and directories located in directory `dname`.
For linux, this includes the `.` and `..` directories.
This can be used in ***for*** expression as shown below:

```
for f in paths.files(".") do
   print(f)
end
```

Optional argument `include` is either a function or a string used to
determine which files are to be included. The function takes the filename
as argument and should return true if the file is to be included.
When a string is provided, the following function is used :

```
function(file)
   return file:find(f)
end
```

Files and directories of sub-folders aren't included.

paths.iterdirs(dname)

Returns an iterator over the directories located in directory `dname`.
This can be used in ***for*** expression as shown below:

```
for dir in paths.iterdirs(".") do
   print(dir)
end
```

Directories of sub-folders, and the `.` and `..` folders aren't included.

paths.iterfiles(dname)

Returns an iterator over the files (non-directories) located in directory `dname`.
This can be used in ***for*** expression as shown below:

```
for file in paths.iterfiles(".") do
   print(file)
end
```

Files of sub-folders, and the `.` and `..` folders aren't included.

paths.mkdir(s)

Create a directory.
Returns `true` on success.

paths.rmdir(s)

Delete an empty directory.
Returns `true` on success.

paths.rmall(s, y)

Recursively delete file or directory `s` and its contents.

Argument `y` must be string `"yes"`
Returns `true` on success.