操作文件和目录

1111次阅读

如果我们要操作文件、目录，可以在命令行下面输入操作系统提供的各种命令来完成。比如dir、cp等命令。

如果要在Python程序中执行这些目录和文件的操作怎么办？其实操作系统提供的命令只是简单地调用了操作系统提供的接口函数，Python内置的os模块也可以直接调用操作系统提供的接口函数。

打开Python交互式命令行，我们来看看如何使用os模块的基本功能：

```
>>> import os
>>> os.name # 操作系统名字
'posix'
```

如果是posix，说明系统是Linux、Unix或Mac OS X，如果是nt，就是Windows系统。

要获取详细的系统信息，可以调用uname()函数：

```
>>> os.uname()
('Darwin', 'iMac.local', '13.3.0', 'Darwin Kernel Version 13.3.0: Tue Jun  3 21:27:35 PDT 2014; root:xnu-2422.110.17~1/RELEASE_X86_64', 'x86_64')
```

环境变量

在操作系统中定义的环境变量，全部保存在os.environ这个dict中，可以直接查看：

```
>>> os.environ
{'VERSIONER_PYTHON_PREFER_32_BIT': 'no', 'TERM_PROGRAM_VERSION': '326', 'LOGNAME': 'michael', 'USER': 'michael', 'PATH': '/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/b
```

要获取某个环境变量的值，可以调用os.getenv()函数：

```
>>> os.getenv('PATH')
'/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/opt/X11/bin:/usr/local/mysql/bin'
```

操作文件和目录

操作文件和目录的函数一部分放在os模块中，一部分放在os.path模块中，这一点要注意一下。查看、创建和删除目录可以这么调用：

```
# 查看当前目录的绝对路径:
>>> os.path.abspath('.')
'/Users/michael'
# 在某个目录下创建一个新目录，
# 首先把新目录的完整路径表示出来:
>>> os.path.join('/Users/michael', 'testdir')
'/Users/michael/testdir'
# 然后创建一个目录:
>>> os.mkdir('/Users/michael/testdir')
# 删掉一个目录:
>>> os.rmdir('/Users/michael/testdir')
```

把两个路径合成一个时，不要直接拼字符串，而要通过os.path.join()函数，这样可以正确处理不同操作系统的路径分隔符。在Linux/Unix/Mac下，os.path.join()返回这样的字符串：

```
part-1/part-2
```

而Windows下会返回这样的字符串：

```
part-1\part-2
```

同样的道理，要拆分路径时，也不要直接去拆字符串，而要通过os.path.split()函数，这样可以把一个路径拆分为两部分，后一部分总是最后级别的目录或文件名：

```
>>> os.path.split('/Users/michael/testdir/file.txt')
('/Users/michael/testdir', 'file.txt')
```

os.path.splitext()可以直接让你得到文件扩展名，很多时候非常方便：

```
>>> os.path.splitext('/path/to/file.txt')
('/path/to/file', '.txt')
```

这些合并、拆分路径的函数并不要求目录和文件要真实存在，它们只对字符串进行操作。

文件操作使用下面的函数。假定当前目录下有一个test.txt文件：

```
# 对文件重命名:
>>> os.rename('test.txt', 'test.py')
# 删掉文件:
>>> os.remove('test.py')
```

但是复制文件的函数居然在os模块中不存在！原因是复制文件并非由操作系统提供的系统调用。理论上讲，我们通过上一节的读写文件可以完成文件复制，只不过要多写很多代码。

幸运的是shutil模块提供了copyfile()的函数，你还可以在shutil模块中找到很多实用函数，它们可以看做是os模块的补充。

最后看看如何利用Python的特性来过滤文件。比如我们要列出当前目录下的所有目录，只需要一行代码：

```
>>> [x for x in os.listdir('.') if os.path.isdir(x)]
['.lein', '.local', '.m2', '.npm', '.ssh', '.Trash', '.vim', 'Adlm', 'Applications', 'Desktop', ...]
```

要列出所有的.py文件，也只需一行代码：

```
>>> [x for x in os.listdir('.') if os.path.isfile(x) and os.path.splitext(x)[1]=='.py']
['apis.py', 'config.py', 'models.py', 'pymonitor.py', 'test_db.py', 'urls.py', 'wsgiapp.py']
```

是不是非常简洁？

小结

Python的os模块封装了操作系统的目录和文件操作，要注意这些函数有的在os模块中，有的在os.path模块中。

练习：编写一个search(s)的函数，能在当前目录以及当前目录的所有子目录下查找文件名包含指定字符串的文件，并打印出完整路径：

```
$ python search.py test
unit_test.log
py/test.py
py/test_os.py
my/logs/unit-test-result.txt
```
