13.9 通过文件名查找文件¶

问题¶

你需要写一个涉及到文件查找操作的脚本，比如对日志归档文件的重命名工具， 你不想在Python脚本中调用shell，或者你要实现一些shell不能做的功能。

解决方案¶

查找文件，可使用 `os.walk()` 函数，传一个顶级目录名给它。 下面是一个例子，查找特定的文件名并答应所有符合条件的文件全路径：

```
#!/usr/bin/env python3.3
import os

def findfile(start, name):
    for relpath, dirs, files in os.walk(start):
        if name in files:
            full_path = os.path.join(start, relpath, name)
            print(os.path.normpath(os.path.abspath(full_path)))

if __name__ == '__main__':
    findfile(sys.argv[1], sys.argv[2])
```

保存脚本为文件findfile.py，然后在命令行中执行它。 指定初始查找目录以及名字作为位置参数，如下：

讨论¶

`os.walk()` 方法为我们遍历目录树， 每次进入一个目录，它会返回一个三元组，包含相对于查找目录的相对路径，一个该目录下的目录名列表， 以及那个目录下面的文件名列表。

对于每个元组，只需检测一下目标文件名是否在文件列表中。如果是就使用 `os.path.join()` 合并路径。 为了避免奇怪的路径名比如 `././foo//bar` ，使用了另外两个函数来修正结果。 第一个是 `os.path.abspath()` ,它接受一个路径，可能是相对路径，最后返回绝对路径。 第二个是 `os.path.normpath()` ，用来返回正常路径，可以解决双斜杆、对目录的多重引用的问题等。

尽管这个脚本相对于UNIX平台上面的很多查找来讲要简单很多，它还有跨平台的优势。 并且，还能很轻松的加入其他的功能。 我们再演示一个例子，下面的函数打印所有最近被修改过的文件：

```
#!/usr/bin/env python3.3

import os
import time

def modified_within(top, seconds):
    now = time.time()
    for path, dirs, files in os.walk(top):
        for name in files:
            fullpath = os.path.join(path, name)
            if os.path.exists(fullpath):
                mtime = os.path.getmtime(fullpath)
                if mtime > (now - seconds):
                    print(fullpath)

if __name__ == '__main__':
    import sys
    if len(sys.argv) != 3:
        print('Usage: {} dir seconds'.format(sys.argv[0]))
        raise SystemExit(1)

    modified_within(sys.argv[1], float(sys.argv[2]))
```

在此函数的基础之上，使用os,os.path,glob等类似模块，你就能实现更加复杂的操作了。 可参考5.11小节和5.13小节等相

关章节。