# Python OS模块学习

os 模块提供了一个统一的操作系统接口函数, 这些接口函数通常是平台指定的,os 模块能在不同操作系统平台如 nt 或 posix中的特定函数间自动切换,从而能实现跨平台操作

## 1.文件操作

build-in函数 open 实现文件创建, 打开, 修改文件的操作

import os

import string

def replace(file, search\_for, replace\_with):

# replace strings in a text file

back = os.path.splitext(file)[0] + ".bak"

temp = os.path.splitext(file)[0] + ".tmp"

try:

# remove old temp file, if any

os.remove(temp)

except os.error:

pass

fi = open(file) #

fo = open(temp, "w")        #

for s in fi.readlines():

fo.write(string.replace(s, search\_for, replace\_with))

fi.close()

fo.close()

try:

# remove old backup file, if any

os.remove(back)

except os.error:

pass

# rename original to backup...

os.rename(file, back)

# ...and temporary to original

os.rename(temp, file)

# try it out!

file = "c:\samples\sample.txt"

replace(file, "hello", "tjena")# search for the string 'hello' and replace with 'tjena

replace(file, "tjena", "hello")

## 2. 目录操作

os 模块包含了许多对目录操作的函数

listdir 函数返回给定目录下的所有文件(包括目录)

import os

for file in os.listdir("c:\qtest"):

print file

getdir 获取当前目录

chdir 改变当前路径

cwd = os.getcwd()

print "1", cwd

# go down

os.chdir("c:\qtest")

print "2", os.getcwd()

# go back up

os.chdir(os.pardir)#返回当前目录的父目录

print "3", os.getcwd()

makedirs removedirs 生成和删除目录

makedirs可以生成多层递归目录, removedirs可以删除多层递归的空目录,若目录中有文件则无法删除

import os

os.makedirs("c:\\test\\multiple\\levels")

fp = open("c:[\\test\\multiple\\levels\\file.txt](file:///\\test\multiple\levels\file.txt)", "w")

fp.write("inspector praline")

fp.close()

# remove the file

os.remove("c:\\test\\multiple\\levels\\file.txt")

# and all empty directories above it

os.removedirs("c:\\test\\multiple\\levels")

mkdir 和 rmdir只能处理单级目录操作.

若要删除非空目录, 可使用 shutil模块中的rmtree函数

## 3. 文件属性的操作

import os

import time

file = 'c:\qtest\editor.pyc'

st = os.stat(file)

print "state", file

def dump(st):

mode, ino, dev, nlink, uid, gid, size, atime, mtime, ctime = st

print "- size:", size, "bytes"

print "- owner:", uid, gid

print "- created:", time.ctime(ctime)

print "- last accessed:", time.ctime(atime)

print "- last modified:", time.ctime(mtime)

print "- mode:", oct(mode)

print "- inode/dev:", ino, dev

print dir(st)

print

dump(st)

# print

fp = open(file)

st = os.fstat(fp.fileno())

print "fstat", file

dump(st)

remark: os.stat(path/file)返回文件的对应属性值st\_mode (protection bits), st\_ino (inode number), st\_dev (device), st\_nlink (number of hard links), st\_uid (user id of owner), st\_gid (group id of owner), st\_size (size of file, in bytes), st\_atime (time of most recent access), st\_mtime (time of most recent content modification), st\_ctime (platform dependent; time of most recent metadata change on Unix, or the time of creation on Windows):

os.fstat(path/file)

Return status for file descriptor *fd*, like [stat()](http://mce_host/wayneye/admin/mk:@MSITStoreC/Documents%20and%20Settings/Administrator/桌面/python265.chm::).

## 4.进程的相关处理

system( )给当前进程输入系统shell命令

import os  
if os.name == "nt":  
      command = "dir"  
else:  
      command = "ls -l"  
      os.system(command)

execvp 开始一个新进程, 以取代目前进程

import os  
import sys  
program = "python"  
arguments = ["hello.py"]  
print os.execvp(program, (program,) + tuple(arguments))  
print "goodbye"

在windows 平台下用spawn( )等函数执行新进程

import os  
import string  
def run(program, \*args):  
# find executable  
for path in string.split(os.environ["PATH"], os.pathsep):  
       file = os.path.join(path, program) + ".exe"  
try:  
       return os.spawnv(os.P\_WAIT, file, (file,) + args)  
except os.error:  
       pass  
raise os.error, "cannot find executable"  
run("python", "hello.py")  
print "goodbye"