

# shutil模块

文件拷贝：使用打开2个文件对象，源文件读取内容，写入目标文件中来完成拷贝过程。但是这样丢失stat数据信息（权限等），因为根本没有复制这些信息过去。

目录复制又怎么办呢？

Python提供了一个方便的库shutil（高级文件操作）。

## copy 复制

```
copyfileobj(fsrc, fdst[, length])
```

文件对象的复制，fsrc和fdst是open打开的文件对象，复制内容。fdst要求可写。  
length 指定了表示buffer的大小；

```
import shutil

with open('o:/test', 'r+') as f1:
    f1.write('abcd\n1234')
    f1.flush()
    with open('o:/test1', 'w+') as f2:
        shutil.copyfileobj(f1, f2) # 可以复制内容吗？为什么，怎么改？
```

```
copyfile(src, dst, *, follow_symlinks=True)
```

复制文件内容，不含元数据。src、dst为文件的路径字符串

本质上调用的就是copyfileobj，所以不带元数据二进制内容复制。

```
copymode(src, dst, *, follow_symlinks=True)
```

仅仅复制权限。

```
shutil.copymode('test1', 'test')

os.stat('test1')
os.stat_result(st_mode=33024, st_ino=3419356, st_dev=64768, st_nlink=1, st_uid=500, st_gid=500,
st_size=0, st_atime=1508722236, st_mtime=1508692014, st_ctime=1508751820)
os.stat('test')
os.stat_result(st_mode=33024, st_ino=3407875, st_dev=64768, st_nlink=1, st_uid=500, st_gid=500,
st_size=3, st_atime=1508690220, st_mtime=1508690177, st_ctime=1508752356)
```

```
copystat(src, dst, *, follow_symlinks=True)
```

复制元数据，stat包含权限

```
(magedu353) [python@nodex cmdb]$ stat test
  File: `test'
  Size: 3          Blocks: 8          IO Block: 4096   regular file
Device: fd00h/64768d    Inode: 3407875    Links: 1
Access: (0777/-rwxrwxrwx)  Uid: ( 500/  python)   Gid: ( 500/  python)
Access: 2017-10-23 00:37:00.181996299 +0800
Modify: 2017-10-23 00:36:17.556997676 +0800
Change: 2017-10-23 17:59:37.415999863 +0800

shutil.copystat('test1','test')

(magedu353) [python@nodex cmdb]$ stat test1
  File: `test1'
  Size: 0          Blocks: 0          IO Block: 4096   regular empty file
Device: fd00h/64768d    Inode: 3419356    Links: 1
Access: (0400/-r-----)  Uid: ( 500/  python)   Gid: ( 500/  python)
Access: 2017-10-23 09:30:36.433991395 +0800
Modify: 2017-10-23 01:06:54.324999197 +0800
Change: 2017-10-23 17:43:40.122993240 +0800

(magedu353) [python@nodex cmdb]$ stat test
  File: `test'
  Size: 3          Blocks: 8          IO Block: 4096   regular file
Device: fd00h/64768d    Inode: 3407875    Links: 1
Access: (0400/-r-----)  Uid: ( 500/  python)   Gid: ( 500/  python)
Access: 2017-10-23 09:30:36.433991395 +0800
Modify: 2017-10-23 01:06:54.324999197 +0800
Change: 2017-10-23 18:06:50.306999032 +0800
```

```
copy(src, dst, *, follow_symlinks=True)
```

复制文件内容、权限和部分元数据，不包括创建时间和修改时间。

本质上调用的是

```
copyfile(src, dst, follow_symlinks=follow_symlinks)
```

```
copymode(src, dst, follow_symlinks=follow_symlinks)
```

`copy2` 比`copy`多了复制全部元数据，但需要平台支持。

本质上调用的是

```
copyfile(src, dst, follow_symlinks=follow_symlinks)
```

```
copystat(src, dst, follow_symlinks=follow_symlinks)
```

```
copytree(src, dst, symlinks=False, ignore=None, copy_function=copy2,
ignore_dangling_symlinks=False)
```

递归复制目录。默认使用`copy2`，也就是带更多的元数据复制。

`src`、`dst`必须是目录，`src`必须存在，`dst`必须**不存在**

ignore = func , 提供一个callable(src, names) -> ignored\_names。提供一个函数, 它会被调用。src是源目录, names是os.listdir(src)的结果, 就是列出src中的文件名, 返回值是要被过滤的文件名的set类型数据。

```
# o:/temp下有a、b目录
def ignore(src, names):
    ig = filter(lambda x: x.startswith('a'), names) # 忽略a
    return set(ig)

shutil.copytree('o:/temp', 'o:/tt/o', ignore=ignore)
```

## rm 删除

shutil.rmtree(path, ignore\_errors=False, onerror=None)

递归删除。如同rm -rf一样危险, 慎用。

它不是原子操作, 有可能删除错误, 就会中断, 已经删除的就删除了。

ignore\_errors为true, 忽略错误。当为False或者omitted时onerror生效。

onerror为callable, 接受函数function、path和execinfo。

```
shutil.rmtree('O:/tmp') # 类似 rm -rf
```

## move 移动

move(src, dst, copy\_function=copy2)

递归移动文件、目录到目标, 返回目标。

本身使用的是 os.rename方法。

如果不支持rename, 如果是目录则copytree再删除源目录。

默认使用copy2方法。

```
os.rename('o:/t.txt', 'o:/temp/t')
os.rename('test3', '/tmp/py/test300')
```

shutil还有打包功能。生成tar并压缩。支持zip、gz、bz、xz。