python 模块介绍- shutil 高级文件操作

目录

项目简介

Python 中文库 https://bitbucket.org/xurongzhong/python-chinese-library 主要基于个人的使用经验，收集一些重要的外部和内部模块的中文教程和实例。发起人是 ouyangchongwu#gmail.com，xurongzhong#gmail.com 。

欢迎大家加入分享经验。联系方法：xurongzhong#gmail.com，微博：http://weibo.com/cizhenshi，python 及测试开发 qq 群 1：113938272，群 2:6089740。

文件下载：

1， https://bitbucket.org/xurongzhong/python-chinese-library/downloads。 推荐

2， hg clone 克隆所有文件 hg clone https://bitbucket.org/xurongzhong/python-chinese-library。

3， https://bitbucket.org/xurongzhong/python-chinese-library/src 浏览文件，右键点击文件，选另存为下载。

Bug 提交：https://bitbucket.org/xurongzhong/python-chinese-library/issuest。

版本管理

版本号	修订发布时间	修订人	备注
V1.0	2013-12-09	Ouyangchongwu#gmail.com	初始版本， 由《The Python Standard Library by Example 2011》和 http://docs.python.org/2/library/shutil.html

			生成。

参考资料：

官方网址： http://docs.python.org/2/library/shutil.html

简介：

功能：高级文件操作。

月下载量：

Python 版本：*Python 1.4 以上。*

当前版本：

下载地址：

平台：跨平台

相关模块：

os 标准模块。

zipfile 标准模块。

tarfile 标准模块

shutil 模块提供了大量的文件的高级操作。特别针对文件拷贝和删除，主要功能为目录和文件操作以及压缩操作。对单个文件的操作也可参见 os 模块。

注意即便是更高级别的文件复制函数（shutil.copy（），shutil.copy2（））也不能复制所有文件的元数据。这意味着在 POSIX 平台上，文件的所有者和组以及访问控制列表都将丢失。在 Mac OS 中资源 fork 和其他元数据无法使用。这意味着资源将丢失，文件类型和创建者代码将不正确。在 Windows 上，文件所有者，ACL 和备用数据流不会被复制。

拷贝文件

shutil.copyfile(*src*, *dst*)：复制文件内容（不包含元数据）从 *src 到 dst*。 DST 必须是完整的目标文件名;拷贝目录参见 shutil.copy()。如果 *src* 和 *dst* 是同一文件，就会引发错

误 shutil.Error。dst 必须是可写的，否则将引发异常 IOError。如果 dst 已经存在，它会被替换。特殊文件，例如字符或块设备和管道不能使用此功能，因为 copyfile 会打开并阅读文件。 src 和 dst 的是字符串形式的路径名。

```
from shutil import *
from glob import glob

print 'BEFORE:', glob('shutil_copyfile.*')
copyfile('shutil_copyfile.py', 'shutil_copyfile.py.copy')
print 'AFTER:', glob('shutil_copyfile.*')
```

执行结果：

```
# python shutil_copyfile.py

BEFORE: ['shutil_copyfile.py']

AFTER: ['shutil_copyfile.py', 'shutil_copyfile.py.copy']

# python shutil_copyfile.py

BEFORE: ['shutil_copyfile.py', 'shutil_copyfile.py.copy']
```

copyfile()调用了底函数层 copyfileobj()。

shutil.copyfileobj(fsrc, fdst[, length])：复制文件内容（不包含元数据）从类文件对象 *src 到*类文件对 *dst*。可选参数 length 指定缓冲区的大小，负数表示一次性读入。默认会把数据切分成小块拷贝，以免占用太多内存。注意：拷贝是从 *fsrc 的当前文件开始*。

```
from shutil import *
import os
from StringIO import StringIO
import sys

class VerboseStringIO(StringIO):
    def read(self, n=-1):
        next = StringIO.read(self, n)
        print 'read(%d) bytes' % n
        return next

lorem_ipsum = '''Lorem ipsum dolor sit amet, consectetuer adipiscing
elit.  Vestibulum aliquam mollis dolor. Donec vulputate nunc ut diam.
Ut rutrum mi vel sem. Vestibulum ante ipsum.'''

print 'Default:'
input = VerboseStringIO(lorem_ipsum)
output = StringIO()
```

```
    copyfileobj(input, output)

    print

    print 'All at once:'
    input = VerboseStringIO(lorem_ipsum)
    output = StringIO()
    copyfileobj(input, output, -1)

    print

    print 'Blocks of 256:'
    input = VerboseStringIO(lorem_ipsum)
    output = StringIO()
    copyfileobj(input, output, 256)
```

执行结果：

```
# python shutil_copyfileobj.py

Default:

read(16384) bytes

read(16384) bytes



All at once:

read(-1) bytes

read(-1) bytes



Blocks of 256:

read(256) bytes

read(256) bytes
```

shutil.copy(src, dst)：复制文件 src 到文件或目录 dst。如果 dst 是目录，使用 src 相同的文件名创建（或覆盖），权限位也会复制。src 和 dst 的是字符串形式的路径名。

```
    from shutil import *
```

```
import os

os.mkdir('example')
print 'BEFORE:', os.listdir('example')
copy('shutil_copy.py', 'example')
print 'AFTER:', os.listdir('example')
```

执行结果：

```
# python shutil_copy.py

BEFORE: []

AFTER: ['shutil_copy.py'
```

shutil.copy2(src, dst)：类似 shutil.copy，元数据也复制，实际上先调用 shutil.copy，然后使用 copystat。这类似于 Unix 命令 cp -p。

```
from shutil import *
import os
import time

def show_file_info(filename):
    stat_info = os.stat(filename)
    print '\tMode    :', stat_info.st_mode
    print '\tCreated :', time.ctime(stat_info.st_ctime)
    print '\tAccessed:', time.ctime(stat_info.st_atime)
    print '\tModified:', time.ctime(stat_info.st_mtime)

os.mkdir('example')
print 'SOURCE:'
show_file_info('shutil_copy2.py')
copy2('shutil_copy2.py', 'example')
print 'DEST:'
show_file_info('example/shutil_copy2.py')
```

执行结果：

```
# python shutil_copy2.py

SOURCE:

        Mode      : 33279

        Created : Fri Dec   6 10:45:52 2013

        Accessed: Fri Dec   6 11:49:40 2013
```

```
        Modified: Mon Feb 21 01:37:46 2011

DEST:

        Mode      : 33279

        Created : Fri Dec  6 11:51:58 2013

        Accessed: Fri Dec  6 11:49:40 2013

        Modified: Mon Feb 21 01:37:46 2011
```

shutil.ignore_patterns(*patterns) 为 copytree 的辅助函数，提供 glob 功能，示例：

```
from shutil import copytree, ignore_patterns

copytree(source, destination, ignore=ignore_patterns('*.pyc', 'tmp*'))
```

拷贝文件元数据

当由 UNIX 下创建文件默认基于 umask 设置权限，copymode()可以复制权限。

shutil.copymode(src, dst)：从 SRC 复制权限位到 DST。该文件的内容，所有者和组不受影响。src 和 dst 的是字符串形式的路径名。

```
from shutil import *
from commands import *
import os

with open('file_to_change.txt', 'wt') as f:
    f.write('content')
os.chmod('file_to_change.txt', 0444)

print 'BEFORE:'
print getstatus('file_to_change.txt')
copymode('shutil_copymode.py', 'file_to_change.txt')
print 'AFTER :'
print getstatus('file_to_change.txt')
```

执行结果：

```
# ./shutil_copymode.py
```

BEFORE:

-r--r--r-- 1 root root 7 Dec 7 17:35 file_to_change.txt

AFTER :

-rwxrwxrwx 1 root root 7 Dec 7 17:35 file_to_change.txt

要想拷贝文件时间戳，需要 copystat。

shutil.copystat(src, dst): 从 src 复制权限位，最后访问时间，最后修改时间，flag 到 dst。该文件的内容，所有者和组不受影响。 src 和 dst 的是给定的字符串路径名。

```
from shutil import *
import os
import time

def show_file_info(filename):
    stat_info = os.stat(filename)
    print '\tMode    :', stat_info.st_mode
    print '\tCreated :', time.ctime(stat_info.st_ctime)
    print '\tAccessed:', time.ctime(stat_info.st_atime)
    print '\tModified:', time.ctime(stat_info.st_mtime)

with open('file_to_change.txt', 'wt') as f:
    f.write('content')
os.chmod('file_to_change.txt', 0444)

print 'BEFORE:'
show_file_info('file_to_change.txt')
copystat('shutil_copystat.py', 'file_to_change.txt')
print 'AFTER:'
show_file_info('file_to_change.txt')
```

执行结果：

```
# python shutil_copystat.py

BEFORE:

        Mode    : 33060

        Created : Mon Dec  9 10:07:26 2013

        Accessed: Sat Dec  7 17:35:08 2013

        Modified: Mon Dec  9 10:07:26 2013
```

AFTER:

Mode : 33279

Created : Mon Dec 9 10:07:26 2013

Accessed: Mon Dec 9 10:06:14 2013

Modified: Mon Feb 21 01:37:46 2011

压缩解压

2.7 以后的版本提供了压缩和解压功能。

shutil.make_archive(base_name, format[, root_dir[, base_dir[, verbose[, dry_run[, owner[, group[, logger]]]]]]])：创建归档文件（如 ZIP 或 TAR），返回其名字。base_name 文件名。format *压缩格式*，“zip”, “tar”, “bztar” or “gztar”。root_dir 压缩的根目录。*base_dir 开始压缩的目录。root_dir* 和 *base_dir 默认都是当前目录*。所有者和组默认为当前用户和组；logger 为 logging.Logger 的实例。

shutil.get_archive_formats()：返回支持的格式列表。默认支持：

gztar：gzip 压缩的 tar 文件

bztar：bzip2 格式的 tar 文件

tar：未压缩的 tar 文件

zip：ZIP 文件

In [3]: shutil.get_archive_formats()

Out[3]:

[('bztar', "bzip2'ed tar-file"),

('gztar', "gzip'ed tar-file"),

('tar', 'uncompressed tar file'),

('zip', 'ZIP file')]

通过使用 register_archive_format()可以增加新格式。

shutil.register_archive_format(name, function[, extra_args[, description]])：注册一个文件格式。不常用。

shutil.unregister_archive_format(name)：移除文件格式，不常用。

压缩实例：

```
from shutil import make_archive
import os
archive_name = os.path.expanduser(os.path.join('~', 'myarchive'))
```

```
root_dir = os.path.expanduser(os.path.join('~', '.ssh'))
make_archive(archive_name, 'gztar', root_dir)
```

上面代码会在当前用户目录生成压缩了.ssh 目录的文件。

```
# tar tzvf myarchive.tar.gz

dr-x------ root/root              0 2013-10-22 18:52 ./

-rw------- root/root           1675 2013-06-06 20:01 ./id_rsa

-rw-r--r-- root/root            392 2013-06-06 20:01 ./id_rsa.pub

-r-------- root/root           1185 2013-08-06 09:41 ./authorized_k
```