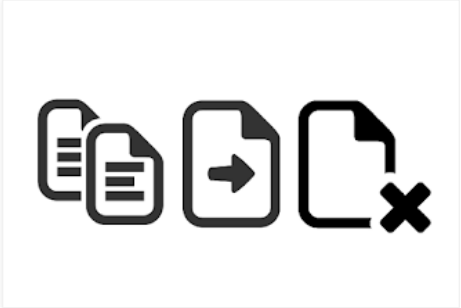
# Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)



**lunes, 19 de octubre de 2015**

## Copiar, mover y borrar archivos/directorios con shutil



El módulo **shutil** consta, entre otras, de funciones para realizar operaciones de alto nivel con archivos y/o directorios. Dentro de las operaciones que se pueden realizar está copiar, mover y borrar archivos y/o directorios; y copiar los permisos y el estado de los archivos.

También, hay otras funciones adicionales para localizar ejecutables, cambiar el propietario y/o el grupo de un archivo/directorio; y consultar información del espacio de un disco.

### Copiar archivos completos o parciales: copyfileobj()

La función **shutil.copyfileobj()** copia el contenido completo de un archivo origen (o sólo una parte del mismo) a un archivo destino. Si el archivo destino no existe será creado y si existe será reemplazado.

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

**Ejemplo: Copia un archivo completo**

```
import shutil, os

ruta = os.getcwd() + os.sep
origen = ruta + 'origen.txt'
destino = ruta + 'destino.txt'

if os.path.exists(origen):
    with open(origen, 'rb') as forigen:
        with open(destino, 'wb') as fdestino:
            shutil.copyfileobj(forigen, fdestino)
            print("Archivo copiado")
```

Esta función también permite copiar un archivo a partir de una posición determinada. Por ejemplo, si se está realizando la lectura secuencial de un archivo y la posición del puntero es distinta de cero se copiará desde la posición actual hasta el final del archivo.

**Ejemplo: Copia contenido de archivo desde el byte número 21**

```
import shutil, os

ruta = os.getcwd() + os.sep
origen = ruta + 'origen.txt'
destino = ruta + 'destino.txt'

if os.path.exists(origen):
    forigen = open(origen,'rb')
    forigen.seek(21)
```

**Buscar**

Buscar

**Python para impacientes**

Python
IPython
EasyGUI
Tkinter
JupyterLab
Numpy

**Anexos**

Guía urgente de MySQL
Guía rápida de SQLite3

**Entradas + populares**

Dar color a las salidas en la consola
En Python para dar color a las salidas en la consola (o en la terminal de texto) existen varias posibilidades. Hay un método basado ...

Instalación de Python, paso a paso
Instalación de Python 3.6 A finales de 2016 se produjo el lanzamiento de Python 3.6 . El propósito de esta entrada es mostrar, pas...

Añadir, consultar, modificar y suprimir elementos en Numpy
Acceder a los elementos de un array. [], [,], ... Acceder a un elemento de un array. Para acceder a un elemento se utiliz...

Variables de control en Tkinter
Variables de control Las variables de control son objetos especiales que se asocian a los widgets para almacenar sus valore...

Cálculo con arrays Numpy
Numpy ofrece todo lo necesario para obtener un buen rendimiento cuando se trata de hacer cálculos con arrays. Por como está concebid...

Operaciones con fechas y horas. Calendarios
Los módulos datetime y calendar amplían las posibilidades del módulo time que provee funciones para manipular expresiones de ti...

Tkinter: interfaces gráficas en Python
Introducción Con Python hay muchas posibilidades para programar una interfaz gráfica de usuario ( GUI ) pero Tkinter es fácil d...

El módulo random
El módulo random de la librería estándar de Python incluye un conjunto de funciones que permiten obtener de distintos modos números a...

Convertir, copiar, ordenar, unir y dividir arrays Numpy
Esta entrada trata sobre algunos métodos que se utilizan en Numpy para convertir listas en arrays y viceversa; para copiar arrays d...

Threading: programación con hilos (I)
En programación, la técnica que permite que una aplicación ejecute

```
    with open(destino, 'wb') as fdestino:
            shutil.copyfileobj(forigen, fdestino)
            print("Archivo copiado")
```

El valor de longitud es opcional y se corresponde con un entero que fija el tamaño del búfer (en bytes) para controlar el consumo de memoria. Este valor afecta a la velocidad de copia, especialmente, de archivos grandes.

**Ejemplo: Copia un archivo ampliando el tamaño del búfer a 256 kb**

```
import shutil, os
from time import time

ruta = os.getcwd() + os.sep
origen = ruta + 'video1.mp4'
destino = ruta + 'video2.mp4'
longitud = 262144

if os.path.exists(origen):
    with open(origen, 'rb') as forigen:
        with open(destino, 'wb') as fdestino:
            tiempo_inicial = time()
            shutil.copyfileobj(forigen, fdestino, longitud)
            tiempo = time() - tiempo_inicial
            print("Video copiado en .2f segundos" % tiempo)
```

## Copiar archivos completos sin metadatos: copyfile()

La función **shutil.copyfile()** copia el contenido completo (sin metadatos) de un archivo origen a un archivo destino, pudiendo estar, uno y otro, en directorios diferentes. Si el archivo destino no existe será creado y si existe será reemplazado.

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

Si el nombre y la ruta del archivo destino coincide con el origen se producirá la excepción **SameFileError**. Por otro lado, si el directorio destino no tiene permisos de escritura se producirá la excepción **OSError**.

Además, si el argumento **follow_symlinks** es **False** y el archivo origen se corresponde con un enlace simbólico se creará otro enlace en el destino.

**Ejemplo: Copia un archivo sin sus metadatos**

```
import shutil, os

ruta = os.getcwd() + os.sep
origen = ruta + 'origen.txt'
destino = ruta + 'destino.txt'

try:
    shutil.copyfile(origen, destino)
    print("Archivo copiado")
except:
    print("Se ha producido un error")
```

## Copiar los permisos de un archivo: copymode()

La función **shutil.copymode()** copia los permisos de un archivo origen a uno destino. Esta copia no afecta al contenido del archivo, ni a la información del propietario y del grupo asignado.

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

Si el argumento **follow_symlinks** es **False** y el origen y el destino son enlaces simbólicos, la función intentará modificar el modo del archivo destino en lugar del archivo al que apunte. Esta funcionalidad no está disponible en todas las plataformas; para más información consultar la función **copystat()**.

**Ejemplo: Copia los permisos de un archivo**

```
import shutil, os, stat

# Antes:
# -r--r--r-- 1 usuario usuario origen.txt
```

simultáneamente varias operaciones en el mismo espacio de proceso se...

**Archivo**

octubre 2015 (2)

**python.org**

**pypi.org**

**Sitios**

- ActivePython
- Anaconda
- Bpython
- Django
- Flask
- Ipython
- IronPython
- Matplotlib
- MicroPython
- Numpy
- Pandas
- Pillow
- PortablePython
- PyBrain
- PyCharm
- PyDev
- PyGame
- Pypi
- PyPy
- Pyramid
- Python.org
- PyTorch
- SciPy.org
- Spyder
- Tensorflow
- TurboGears

```
# -rw-rw-r-- 1 usuario usuario destino.txt

ruta = os.getcwd() + os.sep
origen = ruta + 'origen.txt'
destino = ruta + 'destino.txt'
if os.path.exists(origen) and os.path.exists(destino):
    print('Antes  :', oct(stat.S_IMODE(os.stat(destino).st_mode)))
    shutil.copymode(origen, destino)
    print('Después:', oct(stat.S_IMODE(os.stat(destino).st_mode)))

# Después:
# -r--r--r-- 1 usuario usuario origen.txt
# -r--r--r-- 1 usuario usuario destino.txt
```

En Python 3.3 se agregó el argumento follow_symlinks

## Copiar el estado de un archivo: copystat()

La función **shutil.copystat()** copia los permisos, la fecha-hora del último acceso, la fecha-hora de la última modificación y los atributos de un archivo origen a un archivo destino. Además, en Linux, siempre que sea posible, se copiarán los atributos extendidos.

Esta copia no afecta al contenido del archivo, ni a la información del propietario y del grupo asignado.

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

**Ejemplo: Copia el estado de un archivo (permisos, fechas-horas)**

```
import shutil, os

ruta = os.getcwd() + os.sep
origen = ruta + 'origen.txt'
destino = ruta + 'destino.txt'

# Antes:
# -r--r--r-- 1 usuario usuario 84 oct 13 20:33 origen.txt
# -rw-rw-rw- 1 usuario usuario 63 oct 13 21:45 destino.txt

if os.path.exists(origen) and os.path.exists(destino):
    shutil.copystat(origen, destino)
    print("Estado copiado")

# Después:
# -r--r--r-- 1 usuario usuario 84 oct 13 20:33 origen.txt
# -r--r--r-- 1 usuario usuario 63 oct 13 20:33 destino.txt
```

En Python 3.3 se agregó el argumento **follow_symlinks** y el soporte para los atributos extendidos de Linux

## Copiar archivos con permisos: copy()

La función **shutil.copy()** se utiliza para copiar archivos. Si la cadena indicada en destino es un directorio el archivo se copiará a éste con el mismo nombre del archivo origen.

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

Además, la función después de realizar una copia devuelve el nombre con la ruta del archivo copiado. Hay que tener en cuenta que sólo se copiarán los datos y los permisos del archivo. Otros metadatos como la fecha-hora de creación/modificación no serán copiados. Para incluir también esta información se recomienda el uso de la función **shutil.copy2()**.

Si el argumento **follow_symlinks** es **False** y el origen es un enlace simbólico, en destino se creará un enlace simbólico. Si el argumento **follow_symlinks** es **True** y el origen es un enlace simbólico, en destino se copiará el archivo al que apunte.

**Ejemplo: Copia un archivo (sólo datos y permisos)**

```
import shutil, os

ruta = os.getcwd() + os.sep
origen = ruta + 'origen.txt'
destino = ruta + 'destino.txt'
if os.path.exists(origen):
```

```
    try:
        archivo = shutil.copy(origen, destino)
        print('Copiado...', archivo)
    except:
        print('Error en la copia')
```

En Python 3.3 se agregó el argumento **follow_symlinks**. También, desde esta versión la función devuelve el nombre del archivo copiado.

### Copiar archivos con permisos y metadatos: copy2()

Esta función es idéntica a **shutil.copy()** excepto que **shutil.copy2()** también intenta conservar todos los metadatos de los archivos copiados. Por ello, esta función se apoya en la función **shutil.copystat()**.

```
shutil.copy2(src, dst, *, follow_symlinks=True)
```

**Ejemplo: Copia archivos (datos, permisos y metadatos)**

```
import shutil, os

ruta = os.getcwd() + os.sep
origen = ruta + 'origen.txt'
destino = ruta + 'destino.txt'
if os.path.exists(origen):
    try:
        archivo = shutil.copy2(origen, destino)
        print('Copiado...', archivo)
    except:
        print('Error en la copia')
```

En Python 3.3 se agregó el argumento **follow_symlinks**. También, la posibilidad de copiar los atributos extendidos (Linux) y que la función devuelva la ruta del archivo copiado.

### Copiar un árbol de directorios con sus archivos: copytree()

La función **shutil.copytree()** copia un directorio origen y todo su contenido a un directorio destino que no debe existir. Durante el proceso se copiarán permisos y fechas-horas con **shutil.copystat()**. Además, para copiar los archivos (individuales) se utilizará la función **shutil.copy2()** que incluye permisos y metadatos.

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

**Ejemplo: Copia un árbol de directorios con algunos archivos**

En este ejemplo se utiliza la función **shutil.ignore_patterns()** que permite definir una serie de patrones (como los patrones utilizados con el módulo glob) para que en el proceso de copia se excluyan aquellos ficheros y/o directorios que coincidan con alguno de los patrones expresados.

```
import shutil, os

ruta = os.getcwd() + os.sep
origen = ruta + 'dir1'
destino = ruta + 'dir2'
ignorar_pat = shutil.ignore_patterns('*.dat', 'destino.txt', 'dir11')

if os.path.exists(origen):
    try:
        # Si ignore=None no se excluyen archivos/directorios

        arbol = shutil.copytree(origen, destino, ignore=ignorar_pat)
        print('Árbol copiado a', arbol)
    except:
        print('Error en la copia')
```

Si **symlinks** es **True** los enlaces simbólicos de origen se representan como enlaces simbólicos en el nuevo árbol y los metadatos originales serán copiados cuando la plataforma lo permita. Si **symlinks** es **False** o se omite entonces tanto el contenido como los metadatos de los archivos vinculados se copiarán en el nuevo árbol.

Cuando **symlinks** sea **False** y el archivo apuntado por el enlace simbólico no exista se agregará una excepción a una lista de errores que se genera durante el proceso de copia. Se puede

establecer el indicador opcional **ignore_dangling_symlinks** a **True** para deshabilitar esta excepción.

El argumento copy_function indica la función que se utilizará para copiar que, por defecto, es: **shutil.copy2()**.

### Borrar un árbol de directorios con sus archivos: rmtree()

La función **shutil.rmtree()** elimina un árbol de directorios completo con los archivos existentes. La ruta debe ser de un directorio y no un enlace simbólico que apunte a un directorio.

```
shutil.rmtree(path, ignore_errors=False, onerror=None)
```

**Ejemplo: Borra un árbol de directorios con sus archivos**

```
import shutil, os

ruta = os.getcwd() + os.sep
arbol = ruta + 'directorio'

if os.path.exists(arbol):
    shutil.rmtree(arbol, ignore_errors=False)
    print('Árbol de directorio borrado')
else:
    print('El directorio no existe')
```

Si el argumento **ignore_errors** es **True** los errores producidos por borrados fallidos serán ignorados. Si el valor de este argumento es **False** o se omite los errores producirán una excepción o serán gestionadas por una función especificada en el argumento **onerror**.

### Mover un archivo o un directorio con su contenido: move()

La función **shutil.move()** mueve un archivo o un directorio (y su contenido) a otra ubicación y devuelve la ruta del nuevo destino.

```
shutil.move(src, dst, copy_function=copy2)
```

El argumento opcional **copy_function** se utiliza indicar la función de copia a utilizar (**copy2**, **copy**) en el caso de que no se use la función **os.rename()**. La función **os.rename()** la emplea **shutil.move()** cuando un proceso se puede resolver con renombrar en lugar de copiar.

**Ejemplo: Mover un directorio y su contenido a otro directorio**

```
import shutil, os

ruta = os.getcwd() + os.sep
origen = ruta + 'dir1'
destino = ruta + 'dir2'

if os.path.exists(origen):
    ruta = shutil.move(origen, destino)
    print('El directorio ha sido movido a', ruta)
else:
    print('El directorio origen no existe')
```

### Obtener información del espacio de un disco: disk_usage()

La función **shutil.disk_usage()** devuelve información del espacio del disco donde se encuentre la ruta indicada. La información retorna en una tupla con nombre (namedtuple) con tres valores: espacio total (**total**), usado (**used**) y libre (**free**) expresado en bytes.

```
shutil.disk_usage(path)
```

**Ejemplo: Obtiene información del espacio total, usado y libre, en bytes**

```
import shutil, os

ruta = os.getcwd()
datos = shutil.disk_usage(ruta)
print('Espacio total (bytes): ', datos.total)
print('Espacio usado (bytes): ', datos.used)
print('Espacio libre (bytes): ', datos.free)
```

Nuevo en Python 3.3. y disponible tanto para Unix como para Windows

### Cambiar propietario y/o grupo de un archivo/directorio: chown()

La función **shutil.chown()** permite cambiar el usuario propietario y/o grupo del archivo o directorio indicado en el argumento **path**.

```
shutil.chown(path, user=None, group=None)
```

**Ejemplo: Cambiar el grupo de un archivo**

```
import shutil, os

ruta = os.getcwd() + os.sep
archivo = ruta + 'archivo.html'

# Antes: -rw-rw-r-- 1 usuario www-data archivo.html

try:
    shutil.chown(archivo, group="usuario")
    print('Se ha cambiado el propietario del archivo')

# Después: -rw-rw-r-- 1 usuario usuario  archivo.html

except:
    print('Error')
```

Disponible para Unix/Linux desde Python 3.3

### Obtener la ruta de un archivo ejecutable: which()

La función **which()** del módulo **shutil** permite obtener la ruta de acceso a un ejecutable (o **None** si no hay acceso).

```
shutil.which(cmd, mode=os.F_OK | os.X_OK, path=None)
```

El argumento opcional **mode** permite determinar si un archivo existe y es ejecutable; y el argumento opcional **path** si no se especifica hará que la función **shutil.which()** utilice las rutas de la variable de entorno **PATH** para localizar el ejecutable.

**Ejemplo: Obtener la ruta de Python 3.5**

```
import shutil, os

archivo = 'python3.5'
ruta = shutil.which(archivo)
print(ruta) # /usr/local/bin/python3.5
```

Nuevo en Python 3.3.

Ir al índice del tutorial de Python

Publicado por Pherkad en 14:47

Etiquetas: Python3

Entrada más reciente Inicio Entrada antigua