**Модуль shutil**

Модуль shutil содержит набор функций высокого уровня для обработки файлов, групп файлов, и папок. В частности, доступные здесь функции позволяют копировать, перемещать и удалять файлы и папки. Часто используется вместе с модулем [os](https://pythonworld.ru/moduli/modul-os.html" \t "_blank).

**Операции над файлами и директориями**

**shutil.copyfileobj**(fsrc, fdst[, length]) - скопировать содержимое одного файлового объекта (fsrc) в другой (fdst). Необязательный параметр length - размер буфера при копировании (чтобы весь, возможно огромный, файл не читался целиком в память).

При этом, если позиция указателя в fsrc не 0 (т.е. до этого было сделано что-то наподобие fsrc.read(47)), то будет копироваться содержимое начиная с текущей позиции, а не с начала файла.

**shutil.copyfile**(src, dst, follow\_symlinks=True) - копирует содержимое (но не метаданные) файла src в файл dst. Возвращает dst (т.е. куда файл был скопирован). src и dst это строки - пути к файлам. dst должен быть полным именем файла.

Если src и dst представляют собой один и тот же файл, исключение **shutil.SameFileError**.

Если dst существует, то он будет перезаписан.

Если follow\_symlinks=False и src является ссылкой на файл, то будет создана новая символическая ссылка вместо копирования файла, на который эта символическая ссылка указывает.

**shutil.copymode**(src, dst, follow\_symlinks=True) - копирует права доступа из src в dst. Содержимое файла, владелец, и группа не меняются.

**shutil.copystat**(src, dst, follow\_symlinks=True) - копирует права доступа, время последнего доступа, последнего изменения, и флаги src в dst. Содержимое файла, владелец, и группа не меняются.

**shutil.copy**(src, dst, follow\_symlinks=True) - копирует содержимое файла src в файл **или папку** dst. Если dst является директорией, файл будет скопирован с тем же названием, что было в src. Функция возвращает путь к местонахождению нового скопированного файла.

Если follow\_symlinks=False, и src это ссылка, dst будет ссылкой.

Если follow\_symlinks=True, и src это ссылка, dst будет копией файла, на который ссылается src

copy() копирует содержимое файла, и права доступа.

**shutil.copy2**(src, dst, follow\_symlinks=True) - как copy(), но пытается копировать все метаданные.

**shutil.copytree**(src, dst, symlinks=False, ignore=None, copy\_function=copy2, ignore\_dangling\_symlinks=False) - рекурсивно копирует всё дерево директорий с корнем в src, возвращает директорию назначения.

Директория dst не должна существовать. Она будет создана, вместе с пропущенными родительскими директориями.

Права и времена у директорий копируются copystat(), файлы копируются с помощью функции copy\_function (по умолчанию shutil.copy2()).

Если symlinks=True, ссылки в дереве src будут ссылками в dst, и метаданные будут скопированы настолько, насколько это возможно.

Если False (по умолчанию), будут скопированы содержимое и метаданные файлов, на которые указывали ссылки.

Если symlinks=False, если файл, на который указывает ссылка, не существует, будет добавлено исключение в список ошибок, в исключении shutil.Error в конце копирования.

Можно установить флаг ignore\_dangling\_symlinks=True, чтобы скрыть данную ошибку.

Если ignore не None, то это должна быть функция, принимающая в качестве аргументов имя директории, в которой сейчас copytree(), и список содержимого, возвращаемый os.listdir(). Т.к. copytree()вызывается рекурсивно, ignore вызывается 1 раз для каждой поддиректории. Она должна возвращать список объектов относительно текущего имени директории (т.е. подмножество элементов во втором аргументе). Эти объекты не будут скопированы.

**shutil.ignore\_patterns**(\*patterns) - функция, которая создаёт функцию, которая может быть использована в качестве ignore для copytree(), игнорируя файлы и директории, которые соответствуют [glob](https://pythonworld.ru/moduli/modul-glob.html" \t "_blank)-style шаблонам.

Например:

copytree(source, destination, ignore=ignore\_patterns('\*.pyc', 'tmp\*'))

*# Скопирует все файлы, кроме заканчивающихся на .pyc или начинающихся с tmp*

**shutil.rmtree**(path, ignore\_errors=False, onerror=None) - Удаляет текущую директорию и все поддиректории; path должен указывать на директорию, а не на символическую ссылку.

Если ignore\_errors=True, то ошибки, возникающие в результате неудавшегося удаления, будут проигнорированы. Если False (по умолчанию), эти ошибки будут передаваться обработчику onerror, или, если его нет, то исключение.

На ОС, которые поддерживают функции на основе файловых дескрипторов, по умолчанию используется версия rmtree(), не уязвимая к атакам на символические ссылки.

На других платформах это не так: при подобранном времени и обстоятельствах "хакер" может, манипулируя ссылками, удалить файлы, которые недоступны ему в других обстоятельствах.

Чтобы проверить, уязвима ли система к подобным атакам, можно использовать атрибут rmtree.avoids\_symlink\_attacks.

Если задан onerror, это должна быть функция с 3 параметрами: function, path, excinfo.

Первый параметр, function, это функция, которая создала исключение; она зависит от платформы и интерпретатора. Второй параметр, path, это путь, передаваемый функции. Третий параметр, excinfo - это информация об исключении, возвращаемая sys.exc\_info(). Исключения, вызванные onerror, не обрабатываются.

**shutil.move**(src, dst, copy\_function=copy2) - рекурсивно перемещает файл или директорию (src) в другое место (dst), и возвращает место назначения.

Если dst - существующая директория, то src перемещается внутрь директории. Если dst существует, но не директория, то оно может быть перезаписано.

**shutil.disk\_usage**(path) - возвращает статистику использования дискового пространства как [namedtuple](https://pythonworld.ru/moduli/modul-collections.html" \l "collections-namedtuple" \t "_blank) с арибутами total, used и free, в байтах.

**shutil.chown**(path, user=None, group=None) - меняет владельца и/или группу у файла или директории.

**shutil.which**(cmd, mode=os.F\_OK | os.X\_OK, path=None) - возвращает путь к исполняемому файлу по заданной команде. Если нет соответствия ни с одним файлом, то None. mode это права доступа, требующиеся от файла, по умолчанию ищет только исполняемые.

**Архивация**

Высокоуровневые функции для созданиия и чтения архивированных и сжатых файлов. Основаны на функциях из модулей zipfile и tarfile.

**shutil.make\_archive**(base\_name, format[, root\_dir[, base\_dir[, verbose[, dry\_run[, owner[, group[, logger]]]]]]]) - создаёт архив и возвращает его имя.

base\_name это имя файла для создания, включая путь, но не включая расширения (не нужно писать ".zip" и т.д.).

format - формат архива.

root\_dir - директория (относительно текущей), которую мы архивируем.

base\_dir - директория, в которую будет архивироваться (т.е. все файлы в архиве будут в данной папке).

Если dry\_run=True, архив не будет создан, но операции, которые должны были быть выполнены, запишутся в logger.

owner и group используются при создании tar-архива.

**shutil.get\_archive\_formats**() - список доступных форматов для архивирования.

>>>

**>>>** shutil.get\_archive\_formats()

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

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

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

('xztar', "xz'ed tar-file"),

('zip', 'ZIP file')]

**shutil.unpack\_archive**(filename[, extract\_dir[, format]]) - распаковывает архив. filename - полный путь к архиву.

extract\_dir - то, куда будет извлекаться содержимое (по умолчанию в текущую).

format - формат архива (по умолчанию пытается угадать по расширению файла).

**shutil.get\_unpack\_formats**() - список доступных форматов для разархивирования.

**Запрос размера терминала вывода**

**shutil.get\_terminal\_size**(fallback=(columns, lines)) - возвращает размер окна терминала.

fallback вернётся, если не удалось узнать размер терминала (терминал не поддерживает такие запросы, или программа работает без терминала). По умолчанию (80, 24).

>>>

**>>>** shutil.get\_terminal\_size()

os.terminal\_size(columns=102, lines=29)

**>>>** shutil.get\_terminal\_size() *# Уменьшили окно*

os.terminal\_size(columns=67, lines=17)