

:mod:`shutil` --- High-level file operations

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 1); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 4)

Unknown directive type "module".

```
.. module:: shutil
   :synopsis: High-level file operations, including copying.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 7)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Fred L. Drake, Jr. <fdrake@acm.org>
```

Source code: `:source:`Lib/shutil.py``

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 10); [backlink](#)

Unknown interpreted text role "source".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 12)

Unknown directive type "index".

```
.. index::
   single: file; copying
   single: copying files
```

The `:mod:`shutil`` module offers a number of high-level operations on files and collections of files. In particular, functions are provided which support file copying and removal. For operations on individual files, see also the `:mod:`os`` module.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 18); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 18); [backlink](#)

Unknown interpreted text role "mod".

Warning

Even the higher-level file copying functions (`:func:`shutil.copy``, `:func:`shutil.copy2``) cannot copy all file metadata.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 25); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 25); [backlink](#)

Unknown interpreted text role "func".

On POSIX platforms, this means that file owner and group are lost as well as ACLs. On Mac OS, the resource fork and other metadata are not used. This means that resources will be lost and file type and creator codes will not be correct. On Windows, file owners, ACLs and alternate data streams are not copied.

Directory and files operations

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

main\Doc\library\[cpython-main] [Doc] [library]shutil.rst, line 40)

Unknown directive type "function".

```
.. function:: copyfileobj(fsrc, fdst[, length])
```

Copy the contents of the file-like object *fsrc* to the file-like object *fdst*. The integer *length*, if given, is the buffer size. In particular, a negative *length* value means to copy the data without looping over the source data in chunks; by default the data is read in chunks to avoid uncontrolled memory consumption. Note that if the current file position of the *fsrc* object is not 0, only the contents from the current file position to the end of the file will be copied.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]shutil.rst, line 51)

Unknown directive type "function".

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

Copy the contents (no metadata) of the file named *src* to a file named *dst* and return *dst* in the most efficient way possible. *src* and *dst* are path-like objects or path names given as strings.

dst must be the complete target file name; look at :func:`~shutil.copy` for a copy that accepts a target directory path. If *src* and *dst* specify the same file, :exc:`~SameFileError` is raised.

The destination location must be writable; otherwise, an :exc:`~OSError` exception will be raised. If *dst* already exists, it will be replaced. Special files such as character or block devices and pipes cannot be copied with this function.

If *follow_symlinks* is false and *src* is a symbolic link, a new symbolic link will be created instead of copying the file *src* points to.

```
.. audit-event:: shutil.copyfile src,dst shutil.copyfile
```

```
.. versionchanged:: 3.3
   :exc:`~IOError` used to be raised instead of :exc:`~OSError`.
   Added follow_symlinks argument.
   Now returns dst.
```

```
.. versionchanged:: 3.4
   Raise :exc:`~SameFileError` instead of :exc:`~Error`. Since the former is
   a subclass of the latter, this change is backward compatible.
```

```
.. versionchanged:: 3.8
   Platform-specific fast-copy syscalls may be used internally in order to
   copy the file more efficiently. See
   :ref:`shutil-platform-dependent-efficient-copy-operations` section.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]shutil.rst, line 86)

Unknown directive type "exception".

```
.. exception:: SameFileError
```

This exception is raised if source and destination in :func:`~copyfile` are the same file.

```
.. versionadded:: 3.4
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]shutil.rst, line 94)

Unknown directive type "function".

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

Copy the permission bits from *src* to *dst*. The file contents, owner, and group are unaffected. *src* and *dst* are path-like objects or path names given as strings.

If *follow_symlinks* is false, and both *src* and *dst* are symbolic links, :func:`~copymode` will attempt to modify the mode of *dst* itself (rather than the file it points to). This functionality is not available on every platform; please see :func:`~copystat` for more information. If :func:`~copymode` cannot modify symbolic links on the local platform, and it is asked to do so, it will do nothing and return.

```
.. audit-event:: shutil.copymode src,dst shutil.copymode
```

```
.. versionchanged:: 3.3
   Added follow_symlinks argument.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 111)

Unknown directive type "function".

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

Copy the permission bits, last access time, last modification time, and
flags from *src* to *dst*. On Linux, :func:`copystat` also copies the
"extended attributes" where possible. The file contents, owner, and
group are unaffected. *src* and *dst* are path-like objects or path
names given as strings.

If *follow_symlinks* is false, and *src* and *dst* both
refer to symbolic links, :func:`copystat` will operate on
the symbolic links themselves rather than the files the
symbolic links refer toâ€”reading the information from the
*src* symbolic link, and writing the information to the
*dst* symbolic link.

.. note::

Not all platforms provide the ability to examine and
modify symbolic links. Python itself can tell you what
functionality is locally available.

* If ``os.chmod in os.supports_follow_symlinks`` is
  ``True``, :func:`copystat` can modify the permission
  bits of a symbolic link.

* If ``os.utime in os.supports_follow_symlinks`` is
  ``True``, :func:`copystat` can modify the last access
  and modification times of a symbolic link.

* If ``os.chflags in os.supports_follow_symlinks`` is
  ``True``, :func:`copystat` can modify the flags of
  a symbolic link. (``os.chflags`` is not available on
  all platforms.)

On platforms where some or all of this functionality
is unavailable, when asked to modify a symbolic link,
:func:`copystat` will copy everything it can.
:func:`copystat` never returns failure.

Please see :data:`os.supports_follow_symlinks`
for more information.

.. audit-event:: shutil.copystat src,dst shutil.copystat

.. versionchanged:: 3.3
   Added *follow_symlinks* argument and support for Linux extended attributes.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 158)

Unknown directive type "function".

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

Copies the file *src* to the file or directory *dst*. *src* and *dst*
should be :term:`path-like objects` <path-like object>` or strings. If
*dst* specifies a directory, the file will be copied into *dst* using the
base filename from *src*. Returns the path to the newly created file.

If *follow_symlinks* is false, and *src* is a symbolic link,
*dst* will be created as a symbolic link. If *follow_symlinks*
is true and *src* is a symbolic link, *dst* will be a copy of
the file *src* refers to.

:func:`~shutil.copy` copies the file data and the file's permission
mode (see :func:`os.chmod`). Other metadata, like the
file's creation and modification times, is not preserved.
To preserve all file metadata from the original, use
:func:`~shutil.copy2` instead.

.. audit-event:: shutil.copyfile src,dst shutil.copy

.. audit-event:: shutil.copymode src,dst shutil.copy

.. versionchanged:: 3.3
   Added *follow_symlinks* argument.
   Now returns path to the newly created file.

.. versionchanged:: 3.8
   Platform-specific fast-copy syscalls may be used internally in order to
   copy the file more efficiently. See
   :ref:`shutil-platform-dependent-efficient-copy-operations` section.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 189)

Unknown directive type "function".

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

Identical to :func:`~shutil.copy` except that :func:`~copy2` also attempts to preserve file metadata.

When `*follow_symlinks*` is false, and `*src*` is a symbolic link, :func:`~copy2` attempts to copy all metadata from the `*src*` symbolic link to the newly-created `*dst*` symbolic link. However, this functionality is not available on all platforms. On platforms where some or all of this functionality is unavailable, :func:`~copy2` will preserve all the metadata it can; :func:`~copy2` never raises an exception because it cannot preserve file metadata.

:func:`~copy2` uses :func:`~copystat` to copy the file metadata. Please see :func:`~copystat` for more information about platform support for modifying symbolic link metadata.

```
.. audit-event:: shutil.copyfile src,dst shutil.copy2
```

```
.. audit-event:: shutil.copystat src,dst shutil.copy2
```

```
.. versionchanged:: 3.3
```

Added `*follow_symlinks*` argument, try to copy extended file system attributes too (currently Linux only). Now returns path to the newly created file.

```
.. versionchanged:: 3.8
```

Platform-specific fast-copy syscalls may be used internally in order to copy the file more efficiently. See :ref:`shutil-platform-dependent-efficient-copy-operations` section.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 221)

Unknown directive type "function".

```
.. function:: ignore_patterns(*patterns)
```

This factory function creates a function that can be used as a callable for :func:`~copytree`'s `*ignore*` argument, ignoring files and directories that match one of the glob-style `*patterns*` provided. See the example below.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 228)

Unknown directive type "function".

```
.. function:: copytree(src, dst, symlinks=False, ignore=None, \
    copy_function=copy2, ignore_dangling_symlinks=False, \
    dirs_exist_ok=False)
```

Recursively copy an entire directory tree rooted at `*src*` to a directory named `*dst*` and return the destination directory. `*dirs_exist_ok*` dictates whether to raise an exception in case `*dst*` or any missing parent directory already exists.

Permissions and times of directories are copied with :func:`~copystat`, individual files are copied using :func:`~shutil.copy2`.

If `*symlinks*` is true, symbolic links in the source tree are represented as symbolic links in the new tree and the metadata of the original links will be copied as far as the platform allows; if false or omitted, the contents and metadata of the linked files are copied to the new tree.

When `*symlinks*` is false, if the file pointed by the symlink doesn't exist, an exception will be added in the list of errors raised in an :exc:`~Error` exception at the end of the copy process. You can set the optional `*ignore_dangling_symlinks*` flag to true if you want to silence this exception. Notice that this option has no effect on platforms that don't support :func:`~os.symlink`.

If `*ignore*` is given, it must be a callable that will receive as its arguments the directory being visited by :func:`~copytree`, and a list of its contents, as returned by :func:`~os.listdir`. Since :func:`~copytree` is called recursively, the `*ignore*` callable will be called once for each directory that is copied. The callable must return a sequence of directory and file names relative to the current directory (i.e. a subset of the items in its second argument); these names will then be ignored in the copy process. :func:`~ignore_patterns` can be used to create such a callable that ignores names based on glob-style patterns.

If exception(s) occur, an :exc:`~Error` is raised with a list of reasons.

If `*copy_function*` is given, it must be a callable that will be used to copy each file. It will be called with the source path and the destination path as arguments. By default, :func:`~shutil.copy2` is used, but any function that supports the same signature (like :func:`~shutil.copy`) can be used.

```
.. audit-event:: shutil.copytree src,dst shutil.copytree
```

```
.. versionchanged:: 3.3
```

Copy metadata when `*symlinks*` is false.

```

Now returns *dst*.

.. versionchanged:: 3.2
    Added the *copy_function* argument to be able to provide a custom copy
    function.
    Added the *ignore_dangling_symlinks* argument to silent dangling symlinks
    errors when *symlinks* is false.

.. versionchanged:: 3.8
    Platform-specific fast-copy syscalls may be used internally in order to
    copy the file more efficiently. See
    :ref:`shutil-platform-dependent-efficient-copy-operations` section.

.. versionadded:: 3.8
    The *dirs_exist_ok* parameter.

```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 289)

Unknown directive type "function".

```

.. function:: rmtree(path, ignore_errors=False, onerror=None, *, dir_fd=None)

.. index:: single: directory; deleting

Delete an entire directory tree; *path* must point to a directory (but not a
symbolic link to a directory). If *ignore_errors* is true, errors resulting
from failed removals will be ignored; if false or omitted, such errors are
handled by calling a handler specified by *onerror* or, if that is omitted,
they raise an exception.

This function can support :ref:`paths relative to directory descriptors
<dir_fd>`.

.. note::

    On platforms that support the necessary fd-based functions a symlink
    attack resistant version of :func:`rmtree` is used by default. On other
    platforms, the :func:`rmtree` implementation is susceptible to a symlink
    attack: given proper timing and circumstances, attackers can manipulate
    symlinks on the filesystem to delete files they wouldn't be able to access
    otherwise. Applications can use the :data:`rmtree.avoids_symlink_attacks`
    function attribute to determine which case applies.

If *onerror* is provided, it must be a callable that accepts three
parameters: *function*, *path*, and *excinfo*.

The first parameter, *function*, is the function which raised the exception;
it depends on the platform and implementation. The second parameter,
*path*, will be the path name passed to *function*. The third parameter,
*excinfo*, will be the exception information returned by
:func:`sys.exc_info`. Exceptions raised by *onerror* will not be caught.

.. audit-event:: shutil.rmtree path,dir_fd shutil.rmtree

.. versionchanged:: 3.3
    Added a symlink attack resistant version that is used automatically
    if platform supports fd-based functions.

.. versionchanged:: 3.8
    On Windows, will no longer delete the contents of a directory junction
    before removing the junction.

.. versionchanged:: 3.11
    The *dir_fd* parameter.

.. attribute:: rmtree.avoids_symlink_attacks

    Indicates whether the current platform and implementation provides a
    symlink attack resistant version of :func:`rmtree`. Currently this is
    only true for platforms supporting fd-based directory access functions.

.. versionadded:: 3.3

```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 343)

Unknown directive type "function".

```

.. function:: move(src, dst, copy_function=copy2)

Recursively move a file or directory (*src*) to another location (*dst*)
and return the destination.

If the destination is an existing directory, then *src* is moved inside that
directory. If the destination already exists but is not a directory, it may
be overwritten depending on :func:`os.rename` semantics.

If the destination is on the current filesystem, then :func:`os.rename` is
used. Otherwise, *src* is copied to *dst* using *copy_function* and then
removed. In case of symlinks, a new symlink pointing to the target of *src*
will be created in or as *dst* and *src* will be removed.

```

If `*copy_function*` is given, it must be a callable that takes two arguments `*src*` and `*dst*`, and will be used to copy `*src*` to `*dst*` if `:func:`os.rename`` cannot be used. If the source is a directory, `:func:`copytree`` is called, passing it the `:func:`copy_function``. The default `*copy_function*` is `:func:`copy2``. Using `:func:`~shutil.copy`` as the `*copy_function*` allows the move to succeed when it is not possible to also copy the metadata, at the expense of not copying any of the metadata.

```
.. audit-event:: shutil.move src,dst shutil.move

.. versionchanged:: 3.3
    Added explicit symlink handling for foreign filesystems, thus adapting
    it to the behavior of GNU's :program:`mv`.
    Now returns *dst*.

.. versionchanged:: 3.5
    Added the *copy_function* keyword argument.

.. versionchanged:: 3.8
    Platform-specific fast-copy syscalls may be used internally in order to
    copy the file more efficiently. See
    :ref:`shutil-platform-dependent-efficient-copy-operations` section.

.. versionchanged:: 3.9
    Accepts a :term:`path-like object` for both *src* and *dst*.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 383)

Unknown directive type "function".

```
.. function:: disk_usage(path)

    Return disk usage statistics about the given path as a :term:`named tuple`
    with the attributes *total*, *used* and *free*, which are the amount of
    total, used and free space, in bytes. *path* may be a file or a
    directory.

.. versionadded:: 3.3

.. versionchanged:: 3.8
    On Windows, *path* can now be a file or directory.

.. availability:: Unix, Windows.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 397)

Unknown directive type "function".

```
.. function:: chown(path, user=None, group=None)

    Change owner *user* and/or *group* of the given *path*.

    *user* can be a system user name or a uid; the same applies to *group*. At
    least one argument is required.

    See also :func:`os.chown`, the underlying function.

.. audit-event:: shutil.chown path,user,group shutil.chown

.. availability:: Unix.

.. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 413)

Unknown directive type "function".

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

    Return the path to an executable which would be run if the given *cmd* was
    called. If no *cmd* would be called, return ``None``.

    *mode* is a permission mask passed to :func:`os.access`, by default
    determining if the file exists and executable.

    When no *path* is specified, the results of :func:`os.environ` are used,
    returning either the "PATH" value or a fallback of :attr:`os.defpath`.

    On Windows, the current directory is always prepended to the *path* whether
    or not you use the default or provide your own, which is the behavior the
    command shell uses when finding executables. Additionally, when finding the
    *cmd* in the *path*, the ``PATHEXT`` environment variable is checked. For
    example, if you call ``shutil.which("python")``, :func:`which` will search
    ``PATHEXT`` to know that it should look for ``python.exe`` within the *path*
    directories. For example, on Windows::
```

```
>>> shutil.which("python")
'C:\\Python33\\python.EXE'
```

```
.. versionadded:: 3.3

.. versionchanged:: 3.8
   The :class:`bytes` type is now accepted. If *cmd* type is
   :class:`bytes`, the result type is also :class:`bytes`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 441)

Unknown directive type "exception".

```
.. exception:: Error

   This exception collects exceptions that are raised during a multi-file
   operation. For :func:`copytree`, the exception argument is a list of 3-tuples
   (*srcname*, *dstname*, *exception*).
```

Platform-dependent efficient copy operations

Starting from Python 3.8, all functions involving a file copy (:func:`copyfile`, :func:`~shutil.copy`, :func:`copy2`, :func:`copytree`, and :func:`move`) may use platform-specific "fast-copy" syscalls in order to copy the file more efficiently (see :issue:`33671`). "fast-copy" means that the copying operation occurs within the kernel, avoiding the use of userspace buffers in Python as in "outfd.write(infd.read())".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 452); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 452); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 452); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 452); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 452); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 452); [backlink](#)

Unknown interpreted text role "issue".

On macOS [fcopyfile](#) is used to copy the file content (not metadata).

On Linux :func:`os.sendfile` is used.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 462); [backlink](#)

Unknown interpreted text role "func".

On Windows :func:`shutil.copyfile` uses a bigger default buffer size (1 MiB instead of 64 KiB) and a :func:`memoryview`-based variant of :func:`shutil.copyfileobj` is used.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 464); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 464); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 464); [backlink](#)

Unknown interpreted text role "func".

If the fast-copy operation fails and no data was written in the destination file then `shutil` will silently fallback on using less efficient `:func:`copyfileobj`` function internally.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 468); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 472)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.8
```

copytree example

An example that uses the `:func:`ignore_patterns`` helper:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 479); [backlink](#)

Unknown interpreted text role "func".

```
from shutil import copytree, ignore_patterns

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

This will copy everything except `.pyc` files and files or directories whose name starts with `tmp`.

Another example that uses the `ignore` argument to add a logging call:

```
from shutil import copytree
import logging

def _logpath(path, names):
    logging.info('Working in %s', path)
    return [] # nothing will be ignored

copytree(source, destination, ignore=_logpath)
```

rmtree example

This example shows how to remove a directory tree on Windows where some of the files have their read-only bit set. It uses the `onerror` callback to clear the readonly bit and reattempt the remove. Any subsequent failure will propagate.

```
import os, stat
import shutil

def remove_readonly(func, path, _):
    "Clear the readonly bit and reattempt the removal"
    os.chmod(path, stat.S_IWRITE)
    func(path)

shutil.rmtree(directory, onerror=remove_readonly)
```

Archiving operations

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 525)

Unknown directive type "versionadded".

```
.. versionadded:: 3.2
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 527)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.5
   Added support for the *xztar* format.
```

High-level utilities to create and read compressed and archived files are also provided. They rely on the `mod:`zipfile`` and `mod:`tarfile`` modules.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 531); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 531); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 534)

Unknown directive type "function".

```
.. function:: make_archive(base_name, format, [root_dir, [base_dir, [verbose, [dry_run, [owner, [group, [log

Create an archive file (such as zip or tar) and return its name.

*base_name* is the name of the file to create, including the path, minus
any format-specific extension. *format* is the archive format: one of
"zip" (if the :mod:`zlib` module is available), "tar", "gztar" (if the
:mod:`zlib` module is available), "bztar" (if the :mod:`bz2` module is
available), or "xztar" (if the :mod:`lzma` module is available).

*root_dir* is a directory that will be the root directory of the
archive, all paths in the archive will be relative to it; for example,
we typically chdir into *root_dir* before creating the archive.

*base_dir* is the directory where we start archiving from;
i.e. *base_dir* will be the common prefix of all files and
directories in the archive. *base_dir* must be given relative
to *root_dir*. See :ref:`shutil-archiving-example-with-basedir` for how to
use *base_dir* and *root_dir* together.

*root_dir* and *base_dir* both default to the current directory.

If *dry_run* is true, no archive is created, but the operations that would be
executed are logged to *logger*.

*owner* and *group* are used when creating a tar archive. By default,
uses the current owner and group.

*logger* must be an object compatible with :pep:`282`, usually an instance of
:class:`logging.Logger`.

The *verbose* argument is unused and deprecated.

.. audit-event:: shutil.make_archive base_name,format,root_dir,base_dir shutil.make_archive

.. note::

    This function is not thread-safe.

.. versionchanged:: 3.8
    The modern pax (POSIX.1-2001) format is now used instead of
    the legacy GNU format for archives created with ``format="tar"``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 578)

Unknown directive type "function".

```
.. function:: get_archive_formats()

Return a list of supported formats for archiving.
Each element of the returned sequence is a tuple ``(name, description)``.

By default :mod:`shutil` provides these formats:

- *zip*: ZIP file (if the :mod:`zlib` module is available).
- *tar*: Uncompressed tar file. Uses POSIX.1-2001 pax format for new archives.
- *gztar*: gzip'ed tar-file (if the :mod:`zlib` module is available).
- *bztar*: bzip2'ed tar-file (if the :mod:`bz2` module is available).
- *xztar*: xz'ed tar-file (if the :mod:`lzma` module is available).

You can register new formats or provide your own archiver for any existing
formats, by using :func:`register_archive_format`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 595)

Unknown directive type "function".

```
.. function:: register_archive_format(name, function, [extra_args, [description]])

Register an archiver for the format *name*.

*function* is the callable that will be used to unpack archives. The callable
will receive the *base_name* of the file to create, followed by the
*base_dir* (which defaults to :data:`os.getcwd()`) to start archiving from.
Further arguments are passed as keyword arguments: *owner*, *group*,
*dry_run* and *logger* (as passed in :func:`make_archive`).

If given, *extra_args* is a sequence of ``(name, value)`` pairs that will be
```

used as extra keywords arguments when the archiver callable is used.

description is used by :func:`get_archive_formats` which returns the list of archivers. Defaults to an empty string.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 612)

Unknown directive type "function".

```
.. function:: unregister_archive_format(name)
```

Remove the archive format *name* from the list of supported formats.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 617)

Unknown directive type "function".

```
.. function:: unpack_archive(filename[, extract_dir[, format]])
```

Unpack an archive. *filename* is the full path of the archive.

extract_dir is the name of the target directory where the archive is unpacked. If not provided, the current working directory is used.

format is the archive format: one of "zip", "tar", "gztar", "bztar", or "xztar". Or any other format registered with :func:`register_unpack_format`. If not provided, :func:`unpack_archive` will use the archive file name extension and see if an unpacker was registered for that extension. In case none is found, a :exc:`ValueError` is raised.

```
.. audit-event:: shutil.unpack_archive filename,extract_dir,format shutil.unpack_archive
```

```
.. versionchanged:: 3.7
```

Accepts a :term:`path-like object` for *filename* and *extract_dir*.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 637)

Unknown directive type "function".

```
.. function:: register_unpack_format(name, extensions, function[, extra_args[, description]])
```

Registers an unpack format. *name* is the name of the format and *extensions* is a list of extensions corresponding to the format, like ``.zip`` for Zip files.

function is the callable that will be used to unpack archives. The callable will receive the path of the archive, followed by the directory the archive must be extracted to.

When provided, *extra_args* is a sequence of ``(name, value)`` tuples that will be passed as keywords arguments to the callable.

description can be provided to describe the format, and will be returned by the :func:`get_unpack_formats` function.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 654)

Unknown directive type "function".

```
.. function:: unregister_unpack_format(name)
```

Unregister an unpack format. *name* is the name of the format.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 659)

Unknown directive type "function".

```
.. function:: get_unpack_formats()
```

Return a list of all registered formats for unpacking. Each element of the returned sequence is a tuple ``(name, extensions, description)``.

By default :mod:`shutil` provides these formats:

- *zip*: ZIP file (unpacking compressed files works only if the corresponding module is available).

```
- *tar*: uncompressed tar file.
- *gztar*: gzip'ed tar-file (if the :mod:`zlib` module is available).
- *bz2tar*: bzip2'ed tar-file (if the :mod:`bz2` module is available).
- *xztar*: xz'ed tar-file (if the :mod:`lzma` module is available).
```

You can register new formats or provide your own unpacker for any existing formats, by using :func:`register_unpack_format`.

In this example, we create a gzip'ed tar-file archive containing all files found in the `file:'.ssh'` directory of the user:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 683); *backlink*

Unknown interpreted text role "file".

```
>>> 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, 'gtar', root_dir)
/Users/tarek/myarchive.tar.gz'
```

```

$ tar -tzvf /Users/tarek/myarchive.tar.gz
drwx----- tarek/staff      0 2010-02-01 16:23:40 ./
-rw-r--r-- tarek/staff      609 2008-06-09 13:26:54 ./authorized_keys
-rwxr-xr-x tarek/staff      65 2008-06-09 13:26:54 ./config
-rwx----- tarek/staff     668 2008-06-09 13:26:54 ./id_dsa
-rwxr-xr-x tarek/staff     609 2008-06-09 13:26:54 ./id_dsa.pub
-rw----- tarek/staff    1675 2008-06-09 13:26:54 ./id_rsa
-rw-r--r-- tarek/staff     397 2008-06-09 13:26:54 ./id_rsa.pub
-rw-r--r-- tarek/staff   37192 2010-02-06 18:23:10 ./known_hosts

```

In this example, similar to the [one above](#), we show how to use `func:make_archive`, but this time with the usage of `base_dir`. We now have the following directory structure:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 713); [backlink](#)

Unknown interpreted text role "func".

```
$ tree tmp
tmp
├── â""â"Ĉâ"Ď root
│   └── â""â"Ĉâ"Ď structure
│       ├── â"œâ"Ĉâ"Ď content
│       │   └── â""â"Ĉâ"Ď please_add.txt
│       └── â""â"Ĉâ"Ď do_not_add.txt
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]shutil.rst, line 727); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]shutil.rst, line 727); [backlink](#)

Unknown interpreted text role "file".

```
>>> from shutil import make_archive
>>> import os
>>> archive_name = os.path.expanduser(os.path.join('~', 'myarchive'))
>>> make_archive(
...     archive_name,
...     'tar',
...     root_dir='tmp/root',
...     base_dir='structure/content',
... )
/Users/tarek/my_archive.tar'
```

```
$ python -m tarfile -l /Users/tarek/myarchive.tar
structure/content/
structure/content/please add.txt
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

Unknown directive type "function".

```
.. function:: get_terminal_size(fallback=(columns, lines))
```

Get the size of the terminal window.

For each of the two dimensions, the environment variable, ``COLUMNS`` and ``LINES`` respectively, is checked. If the variable is defined and the value is a positive integer, it is used.

When ``COLUMNS`` or ``LINES`` is not defined, which is the common case, the terminal connected to :data:`sys.__stdout__` is queried by invoking :func:`os.get_terminal_size`.

If the terminal size cannot be successfully queried, either because the system doesn't support querying, or because we are not connected to a terminal, the value given in ``fallback`` parameter is used. ``fallback`` defaults to ``(80, 24)`` which is the default size used by many terminal emulators.

The value returned is a named tuple of type :class:`os.terminal_size`.

See also: The Single UNIX Specification, Version 2, 'Other Environment Variables'.

```
.. versionadded:: 3.3
```

```
.. versionchanged:: 3.11
```

The ``fallback`` values are also used if :func:`os.get_terminal_size` returns zeroes.