11.3. `stat` — Interpreting `stat()` results¶

The `stat` module defines constants and functions for interpreting the results of `os.stat()`, `os.fstat()` and `os.lstat()` (if they exist). For complete details about the `stat()`, `fstat()` and `lstat()` calls, consult the documentation for your system.

The `stat` module defines the following functions to test for specific file types:

`stat.S_ISDIR`(*mode*)¶
Return non-zero if the mode is from a directory.

`stat.S_ISCHR`(*mode*)¶
Return non-zero if the mode is from a character special device file.

`stat.S_ISBLK`(*mode*)¶
Return non-zero if the mode is from a block special device file.

`stat.S_ISREG`(*mode*)¶
Return non-zero if the mode is from a regular file.

`stat.S_ISFIFO`(*mode*)¶
Return non-zero if the mode is from a FIFO (named pipe).

`stat.S_ISLNK`(*mode*)¶
Return non-zero if the mode is from a symbolic link.

`stat.S_ISSOCK`(*mode*)¶
Return non-zero if the mode is from a socket.

Two additional functions are defined for more general manipulation of the file's mode:

`stat.S_IMODE`(*mode*)¶
Return the portion of the file's mode that can be set by `os.chmod()`—that is, the file's permission bits, plus the sticky bit, set-group-id, and set-user-id bits (on systems that support them).

`stat.S_IFMT`(*mode*)¶
Return the portion of the file's mode that describes the file type (used by the `S_IS*()` functions above).

Normally, you would use the `os.path.is*()` functions for testing the type of a file; the functions here are useful when you are doing multiple tests of the same file and wish to avoid the overhead of the `stat()` system call for each test. These are also useful when checking for information about a file that isn't handled by `os.path`, like the tests for block and character devices.

All the variables below are simply symbolic indexes into the 10-tuple returned by `os.stat()`, `os.fstat()` or `os.lstat()`.

`stat.ST_MODE`¶
Inode protection mode.

`stat.ST_INO`¶
Inode number.

`stat.ST_DEV`¶
Device inode resides on.

`stat.ST_NLINK`¶
Number of links to the inode.

`stat.ST_UID`¶
User id of the owner.

`stat.ST_GID`¶
Group id of the owner.

`stat.ST_SIZE`¶
Size in bytes of a plain file; amount of data waiting on some special files.

`stat.ST_ATIME`¶
Time of last access.

`stat.ST_MTIME`¶

Time of last modification.

`stat.ST_CTIME`¶

The “ctime” as reported by the operating system. On some systems (like Unix) is the time of the last metadata change, and, on others (like Windows), is the creation time (see platform documentation for details).

The interpretation of “file size” changes according to the file type. For plain files this is the size of the file in bytes. For FIFOs and sockets under most flavors of Unix (including Linux in particular), the “size” is the number of bytes waiting to be read at the time of the call to `os.stat()`, `os.fstat()`, or `os.lstat()`; this can sometimes be useful, especially for polling one of these special files after a non-blocking open. The meaning of the size field for other character and block devices varies more, depending on the implementation of the underlying system call.

The variables below define the flags used in the `ST_MODE` field.

Use of the functions above is more portable than use of the first set of flags:

`stat.S_IFMT`

Bit mask for the file type bit fields.

`stat.S_IFSOCK`¶

Socket.

`stat.S_IFLNK`¶

Symbolic link.

`stat.S_IFREG`¶

Regular file.

`stat.S_IFBLK`¶

Block device.

`stat.S_IFDIR`¶

Directory.

`stat.S_IFCHR`¶

Character device.

`stat.S_IFIFO`¶

FIFO.

The following flags can also be used in the *mode* argument of `os.chmod()`:

`stat.S_ISUID`¶

Set UID bit.

`stat.S_ISGID`¶

Set-group-ID bit. This bit has several special uses. For a directory it indicates that BSD semantics is to be used for that directory: files created there inherit their group ID from the directory, not from the effective group ID of the creating process, and directories created there will also get the `S_ISGID` bit set. For a file that does not have the group execution bit (`S_IXGRP`) set, the set-group-ID bit indicates mandatory file/record locking (see also `S_ENFMT`).

`stat.S_ISVTX`¶

Sticky bit. When this bit is set on a directory it means that a file in that directory can be renamed or deleted only by the owner of the file, by the owner of the directory, or by a privileged process.

`stat.S_IRWXU`¶

Mask for file owner permissions.

`stat.S_IRUSR`¶

Owner has read permission.

`stat.S_IWUSR`¶

Owner has write permission.

`stat.S_IXUSR`¶

Owner has execute permission.

`stat.S_IRWXG`¶

Mask for group permissions.

`stat.S_IRGRP`¶

Group has read permission.

`stat.S_IWGRP`¶

Group has write permission.

`stat.S_IXGRP`¶

Group has execute permission.

`stat.S_IRWXO`¶

Mask for permissions for others (not in group).

`stat.S_IROTH`¶

Others have read permission.

`stat.S_IWOTH`¶

Others have write permission.

`stat.S_IXOTH`¶

Others have execute permission.

`stat.S_ENFMT`¶

System V file locking enforcement. This flag is shared with `S_ISGID`: file/record locking is enforced on files that do not have the group execution bit (`S_IXGRP`) set.

`stat.S_IREAD`¶

Unix V7 synonym for `S_IRUSR`.

`stat.S_IWRITE`¶

Unix V7 synonym for `S_IWUSR`.

`stat.S_IEXEC`¶

Unix V7 synonym for `S_IXUSR`.

Example:

```
import os, sys
from stat import *

def walktree(top, callback):
   '''recursively descend the directory tree rooted at top,
      calling the callback function for each regular file'''

   for f in os.listdir(top):
       pathname = os.path.join(top, f)
       mode = os.stat(pathname)[ST_MODE]
       if S_ISDIR(mode):
           # It's a directory, recurse into it
           walktree(pathname, callback)
       elif S_ISREG(mode):
           # It's a file, call the callback function
           callback(pathname)
       else:
           # Unknown file type, print a message
           print 'Skipping %s' % pathname

def visitfile(file):
   print 'visiting', file

if __name__ == '__main__':
   walktree(sys.argv[1], visitfile)
```