Next: File Owner, Previous: Reading Attributes, Up: File Attributes [Contents][Index]

14.9.3 Testing the Type of a File

The *file mode*, stored in the `st_mode` field of the file attributes, contains two kinds of information: the file type code, and the access permission bits. This section discusses only the type code, which you can use to tell whether the file is a directory, socket, symbolic link, and so on. For details about access permissions see Permission Bits.

There are two ways you can access the file type information in a file mode. Firstly, for each file type there is a *predicate macro* which examines a given file mode and returns whether it is of that type or not. Secondly, you can mask out the rest of the file mode to leave just the file type code, and compare this against constants for each of the supported file types.

All of the symbols listed in this section are defined in the header file `sys/stat.h`.

The following predicate macros test the type of a file, given the value *m* which is the `st_mode` field returned by `stat` on that file:

Macro: *int* **S_ISDIR** *(mode_t m)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

This macro returns non-zero if the file is a directory.

Macro: *int* **S_ISCHR** *(mode_t m)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

This macro returns non-zero if the file is a character special file (a device like a terminal).

Macro: *int* **S_ISBLK** *(mode_t m)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

This macro returns non-zero if the file is a block special file (a device like a disk).

Macro: *int* **S_ISREG** *(mode_t m)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

This macro returns non-zero if the file is a regular file.

Macro: *int* **S_ISFIFO** *(mode_t m)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

This macro returns non-zero if the file is a FIFO special file, or a pipe. See Pipes and FIFOs.

Macro: *int* **S_ISLNK** *(mode_t m)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

This macro returns non-zero if the file is a symbolic link. See Symbolic Links.

Macro: *int* **S_ISSOCK** *(mode_t m)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

This macro returns non-zero if the file is a socket. See Sockets.

An alternate non-POSIX method of testing the file type is supported for compatibility with BSD. The mode can be bitwise AND-ed with `S_IFMT` to extract the file type code, and compared to the appropriate constant. For example,

```
S_ISCHR (mode)
```

is equivalent to:

```
((mode & S_IFMT) == S_IFCHR)
```

Macro: *int* **S_IFMT**

This is a bit mask used to extract the file type code from a mode value.

These are the symbolic names for the different file type codes:

`S_IFDIR`

This is the file type constant of a directory file.

`S_IFCHR`

This is the file type constant of a character-oriented device file.

`S_IFBLK`

This is the file type constant of a block-oriented device file.

`S_IFREG`

This is the file type constant of a regular file.

`S_IFLNK`

This is the file type constant of a symbolic link.

`S_IFSOCK`

This is the file type constant of a socket.

`S_IFIFO`

This is the file type constant of a FIFO or pipe.

The POSIX.1b standard introduced a few more objects which possibly can be implemented as objects in

the filesystem. These are message queues, semaphores, and shared memory objects. To allow differentiating these objects from other files the POSIX standard introduced three new test macros. But unlike the other macros they do not take the value of the `st_mode` field as the parameter. Instead they expect a pointer to the whole `struct stat` structure.

Macro: *int* **S_TYPEISMQ** *(struct stat *s)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

If the system implements POSIX message queues as distinct objects and the file is a message queue object, this macro returns a non-zero value. In all other cases the result is zero.

Macro: *int* **S_TYPEISSEM** *(struct stat *s)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

If the system implements POSIX semaphores as distinct objects and the file is a semaphore object, this macro returns a non-zero value. In all other cases the result is zero.

Macro: *int* **S_TYPEISSHM** *(struct stat *s)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

If the system implements POSIX shared memory objects as distinct objects and the file is a shared memory object, this macro returns a non-zero value. In all other cases the result is zero.

Next: File Owner, Previous: Reading Attributes, Up: File Attributes [Contents][Index]