NAME

mac_get_file, **mac_get_link**, **mac_get_fd**, **mac_get_peer**, **mac_get_pid**, **mac_get_proc** - get the label of a file, socket, socket peer or process

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/mac.h>

int
mac_get_file(*const char *path*, *mac_t label*);

int
mac_get_link(*const char *path*, *mac_t label*);

int
mac_get_fd(*int fd*, *mac_t label*);

int
mac_get_peer(*int fd*, *mac_t label*);

int
mac_get_pid(*pid_t pid*, *mac_t label*);

int
mac_get_proc(*mac_t label*);

DESCRIPTION

The **mac_get_file**() system call returns the label associated with a file specified by pathname. The **mac_get_link**() function is the same as **mac_get_file**(), except that it does not follow symlinks.

The **mac_get_fd**() system call returns the label associated with an object referenced by the specified file descriptor. Note that in the case of a file system socket, the label returned will be the socket label, which may be different from the label of the on-disk node acting as a rendezvous for the socket. The **mac_get_peer**() system call returns the label associated with the remote endpoint of a socket; the exact semantics of this call will depend on the protocol domain, communications type, and endpoint; typically this label will be cached when a connection-oriented protocol instance is first set up, and is undefined for datagram protocols.

The **mac_get_pid**() and **mac_get_proc**() system calls return the process label associated with an arbitrary process ID, or the current process.

Label storage for use with these calls must first be allocated and prepared using the mac_prepare(3) functions. When an application is done using a label, the memory may be returned using mac_free(3).

ERRORS

[EACCES] A component of *path* is not searchable, or MAC read access to the file is denied.

[EINVAL] The requested label operation is not valid for the object referenced by *fd*.

[ENAMETOOLONG]
The pathname pointed to by *path* exceeds PATH_MAX, or a component of the pathname exceeds NAME_MAX.

[ENOENT] A component of *path* does not exist.

[ENOMEM] Insufficient memory is available to allocate a new MAC label structure.

[ENOTDIR] A component of *path* is not a directory.

SEE ALSO

mac(3), mac_free(3), mac_prepare(3), mac_set(3), mac_text(3), posix1e(3), mac(4), mac(9)

STANDARDS

POSIX.1e is described in IEEE POSIX.1e draft 17. Discussion of the draft continues on the cross-platform POSIX.1e implementation mailing list. To join this list, see the FreeBSD POSIX.1e implementation page for more information.

HISTORY

Support for Mandatory Access Control was introduced in FreeBSD 5.0 as part of the TrustedBSD Project.