## NAME

**getmode**, **setmode** - modify mode bits

## LIBRARY

Standard C Library (libc, -lc)

## SYNOPSIS

**#include <unistd.h>**

*mode_t*
**getmode**(*const void *set*, *mode_t mode*);

*void **
**setmode**(*const char *mode_str*);

## DESCRIPTION

The **getmode**() function returns a copy of the file permission bits *mode* as altered by the values pointed to by *set*. While only the mode bits are altered, other parts of the file mode may be examined.

The **setmode**() function takes an absolute (octal) or symbolic value, as described in chmod(1), as an argument and returns a pointer to mode values to be supplied to **getmode**(). Because some of the symbolic values are relative to the file creation mask, **setmode**() may call umask(2). If this occurs, the file creation mask will be restored before **setmode**() returns. If the calling program changes the value of its file creation mask after calling **setmode**(), **setmode**() must be called again if **getmode**() is to modify future file modes correctly.

If the mode passed to **setmode**() is invalid or if memory cannot be allocated for the return value, **setmode**() returns NULL.

The value returned from **setmode**() is obtained from **malloc**() and should be returned to the system with **free**() when the program is done with it, generally after a call to **getmode**().

## ERRORS

The **setmode**() function may fail and set errno for any of the errors specified for the library routine malloc(3) or strtol(3). In addition, **setmode**() will fail and set *errno* to:

[EINVAL] The *mode* argument does not represent a valid mode.

## SEE ALSO

chmod(1), stat(2), umask(2), malloc(3)

## HISTORY

The **getmode**() and **setmode**() functions first appeared in 4.4BSD.

## BUGS

The **setmode**() function is not thread safe. Files created in other threads while **setmode**() is being called may be created with a umask of 0.