NAME

sysdecode_mask, **sysdecode_accessmode**, **sysdecode_atflags**, **sysdecode_capfcntlrights**, **sysdecode_close_range_flags**, **sysdecode_fcntl_fileflags**, **sysdecode_fileflags**, **sysdecode_filemode**, **sysdecode_flock_operation**, **sysdecode_mlockall_flags**, **sysdecode_mmap_flags**, **sysdecode_mmap_prot**, **sysdecode_mount_flags**, **sysdecode_msg_flags**, **sysdecode_msync_flags**, **sysdecode_open_flags**, **sysdecode_pipe2_flags**, **sysdecode_pollfd_events**, **sysdecode_reboot_howto**, **sysdecode_rfork_flags**, **sysdecode_semget_flags**, **sysdecode_sendfile_flags**, **sysdecode_shmat_flags**, **sysdecode_sctp_nxt_flags**, **sysdecode_sctp_rcv_flags**, **sysdecode_sctp_snd_flags**, **sysdecode_socket_type**, **sysdecode_thr_create_flags**, **sysdecode_umtx_cvwait_flags**, **sysdecode_umtx_rwlock_flags**, **sysdecode_vmprot**, **sysdecode_wait4_options**, **sysdecode_wait6_options** **-** print name of various bitmask values

LIBRARY

System Argument Decoding Library (libsysdecode, -lsysdecode)

SYNOPSIS

#include <sysdecode.h>

bool
sysdecode_access_mode(*FILE *fp, int mode, int *rem*);

bool
sysdecode_atflags(*FILE *fp, int flags, int *rem*);

bool
sysdecode_cap_fcntlrights(*FILE *fp, uint32_t rights, uint32_t *rem*);

bool
sysdecode_close_range_flags(*FILE *fp, int flags, int *rem*);

bool
sysdecode_fcntl_fileflags(*FILE *fp, int flags, int *rem*);

bool
sysdecode_fileflags(*FILE *fp, fflags_t flags, fflags_t *rem*);

bool
sysdecode_filemode(*FILE *fp, int mode, int *rem*);

bool

sysdecode_flock_operation(*FILE *fp*, *int operation*, *int *rem*);

bool
sysdecode_mlockall_flags(*FILE *fp*, *int flags*, *int *rem*);

bool
sysdecode_mmap_flags(*FILE *fp*, *int flags*, *int *rem*);

bool
sysdecode_mmap_prot(*FILE *fp*, *int prot*, *int *rem*);

bool
sysdecode_mount_flags(*FILE *fp*, *int flags*, *int *rem*);

bool
sysdecode_msg_flags(*FILE *fp*, *int flags*, *int *rem*);

bool
sysdecode_msync_flags(*FILE *fp*, *int flags*, *int *rem*);

bool
sysdecode_open_flags(*FILE *fp*, *int flags*, *int *rem*);

bool
sysdecode_pipe2_flags(*FILE *fp*, *int flags*, *int *rem*);

bool
sysdecode_pollfd_events(*FILE *fp*, *int flags*, *int *rem*);

bool
sysdecode_reboot_howto(*FILE *fp*, *int howto*, *int *rem*);

bool
sysdecode_rfork_flags(*FILE *fp*, *int flags*, *int *rem*);

bool
sysdecode_sctp_nxt_flags(*FILE *fp*, *int flags*, *int *rem*);

bool
sysdecode_sctp_rcv_flags(*FILE *fp*, *int flags*, *int *rem*);

bool
sysdecode_sctp_snd_flags(*FILE *fp*, *int flags*, *int *rem*);

bool
sysdecode_semget_flags(*FILE *fp*, *int flags*, *int *rem*);

bool
sysdecode_sendfile_flags(*FILE *fp*, *int flags*, *int *rem*);

bool
sysdecode_shmat_flags(*FILE *fp*, *int flags*, *int *rem*);

bool
sysdecode_socket_type(*FILE *fp*, *int type*, *int *rem*);

bool
sysdecode_thr_create_flags(*FILE *fp*, *int flags*, *int *rem*);

bool
sysdecode_umtx_cvwait_flags(*FILE *fp*, *u_long flags*, *u_long *rem*);

bool
sysdecode_umtx_rwlock_flags(*FILE *fp*, *u_long flags*, *u_long *rem*);

bool
sysdecode_vmprot(*FILE *fp*, *int type*, *int *rem*);

bool
sysdecode_wait4_options(*FILE *fp*, *int options*, *int *rem*);

bool
sysdecode_wait6_options(*FILE *fp*, *int options*, *int *rem*);

DESCRIPTION

The **sysdecode_mask** functions are used to generate a text description of an integer value built from a mask of bitfields. The text description lists the C macros for field values joined by pipe ‘|’ characters matching the format used in C source code. Most of the values decoded by these functions are passed as arguments to system calls, though some of these values are used internally in the kernel.

Each function writes the text description to *fp*. The second argument should contain the integer value to be decoded. The *rem* argument is set to the value of any bits that were not decoded (bit fields that do not have a corresponding C macro). *rem* may be set to NULL if the caller does not need this value. Each function returns true if any bit fields in the value were decoded and false if no bit fields were decoded.

Most of these functions decode an argument passed to a system call:

Function	**System Call**	**Argument**
sysdecode_access_mode()	access(2)	*mode*
sysdecode_atflags()	chflagsat(2), fstatat(2)	*atflag*, *flag*
sysdecode_cap_fcntlrights()	cap_fcntls_limit(2)	*fcntlrights*
sysdecode_fileflags()	chflags(2)	*flags*
sysdecode_filemode()	chmod(2), open(2)	mode
sysdecode_flock_operation()	flock(2)	*operation*
sysdecode_mlockall_flags()	mlockall(2)	*flags*
sysdecode_mmap_flags()	mmap(2)	*flags*
sysdecode_mmap_prot()	mmap(2)	*prot*
sysdecode_mount_flags()	mount(2)	*flags*
sysdecode_msg_flags()	recv(2), send(2)	*flags*
sysdecode_msync_flags()	msync(2)	*flags*
sysdecode_open_flags()	open(2)	*flags*
sysdecode_pipe2_flags()	pipe2	*flags*
sysdecode_reboot_howto()	reboot(2)	*howto*
sysdecode_rfork_flags()	rfork(2)	*flags*
sysdecode_semget_flags()	semget(2)	*flags*
sysdecode_sendfile_flags()	sendfile(2)	*flags*
sysdecode_shmat_flags()	shmat(2)	*flags*
sysdecode_socket_type()	socket(2)	*type*
sysdecode_thr_create_flags()	thr_create(2)	*flags*
sysdecode_wait4_options()	wait4(2)	*options*
sysdecode_wait6_options()	wait6(2)	*options*

Other functions decode the values described below:

sysdecode_fcntl_fileflags()	The file flags used with the F_GETFL and F_SETFL fcntl(2) commands.
sysdecode_pollfd_events()	The *events* and *revents* members of a *struct pollfd*.
sysdecode_sctp_nxt_flags()	The *nxt_flags* member of a *struct sctp_nxtinfo*.

sysdecode_sctp_rcv_flags() The *rcv_flags* member of a *struct sctp_rcvinfo*.

sysdecode_sctp_snd_flags() The *snd_flags* member of a *struct sctp_sndinfo*.

sysdecode_umtx_cvwait_flags()
The *val* argument to _umtx_op(2) for UMTX_OP_CV_WAIT operations.

sysdecode_umtx_rwlock_flags()
The *val* argument to _umtx_op(2) for UMTX_OP_RW_RDLOCK operations.

sysdecode_vmprot() The memory protection flags stored in *vm_prot_t* variables.

RETURN VALUES

The **sysdecode_mask** functions return true if any bit fields in the value were decoded and false if no bit fields were decoded.

SEE ALSO

sysdecode(3), sysdecode_enum(3)