

**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:

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

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)`