The Open Group Base Specifications Issue 7, 2018 edition
IEEE Std 1003.1-2017 (Revision of IEEE Std 1003.1-2008)
Copyright © 2001-2018 IEEE and The Open Group

NAME

sys/socket.h - main sockets header

SYNOPSIS

```
#include <sys/socket.h>
```

DESCRIPTION

The *<sys/socket.h>* header shall define the **socklen_t** type, which is an integer type of width of at least 32 bits; see APPLICATION USAGE.

The *<sys/socket.h>* header shall define the **sa_family_t** unsigned integer type.

The *<sys/socket.h>* header shall define the **sockaddr** structure, which shall include at least the following members:

```
sa_family_t  sa_family  Address family.
char         sa_data[]  Socket address (variable-length data).
```

The **sockaddr** structure is used to define a socket address which is used in the *bind*(), *connect*(), *getpeername*(), *getsockname*(), *recvfrom*(), and *sendto*() functions.

The *<sys/socket.h>* header shall define the **sockaddr_storage** structure, which shall be:

- Large enough to accommodate all supported protocol-specific address structures
- Aligned at an appropriate boundary so that pointers to it can be cast as pointers to protocol-specific address structures and used to access the fields of those structures without alignment problems

The **sockaddr_storage** structure shall include at least the following members:

```
sa_family_t   ss_family
```

When a pointer to a **sockaddr_storage** structure is cast as a pointer to a **sockaddr** structure, the *ss_family* field of the **sockaddr_storage** structure shall map onto the *sa_family* field of the **sockaddr** structure. When a pointer to a **sockaddr_storage** structure is cast as a pointer to a protocol-specific address structure, the *ss_family* field shall map onto a field of that structure that is of type **sa_family_t** and that identifies the protocol's address family.

The *<sys/socket.h>* header shall define the **msghdr** structure, which shall include at least the following members:

```
void          *msg_name        Optional address.
socklen_t      msg_namelen     Size of address.
struct iovec  *msg_iov         Scatter/gather array.
int            msg_iovlen      Members in msg_iov.
void          *msg_control     Ancillary data; see below.
socklen_t      msg_controllen  Ancillary data buffer len.
int            msg_flags       Flags on received message.
```

The **msghdr** structure is used to minimize the number of directly supplied parameters to the *recvmsg()* and *sendmsg()* functions. This structure is used as a *value- result* parameter in the *recvmsg()* function and *value* only for the *sendmsg()* function.

The *<sys/socket.h>* header shall define the **iovec** structure as described in *<sys/uio.h>*.

The *<sys/socket.h>* header shall define the **cmsghdr** structure, which shall include at least the following members:

```
socklen_t  cmsg_len    Data byte count, including the cmsghdr.
int        cmsg_level  Originating protocol.
int        cmsg_type   Protocol-specific type.
```

The **cmsghdr** structure is used for storage of ancillary data object information.

Ancillary data consists of a sequence of pairs, each consisting of a **cmsghdr** structure followed by a data array. The data array contains the ancillary data message, and the **cmsghdr** structure contains descriptive information that allows an application to correctly parse the data.

The values for *cmsg_level* shall be legal values for the *level* argument to the *getsockopt()* and *setsockopt()* functions. The system documentation shall specify the *cmsg_type* definitions for the supported protocols.

Ancillary data is also possible at the socket level. The *<sys/socket.h>* header shall define the following symbolic constant for use as the *cmsg_type* value when *cmsg_level* is SOL_SOCKET:

SCM_RIGHTS
    Indicates that the data array contains the access rights to be sent or received.

The *<sys/socket.h>* header shall define the following macros to gain access to the data arrays in the ancillary data associated with a message header:

CMSG_DATA(*cmsg*)
    If the argument is a pointer to a **cmsghdr** structure, this macro shall return an unsigned character pointer to the data array associated with the **cmsghdr** structure.

CMSG_NXTHDR(*mhdr,cmsg*)
    If the first argument is a pointer to a **msghdr** structure and the second argument is a pointer to a **cmsghdr** structure in the ancillary data pointed to by the *msg_control* field of that **msghdr** structure, this macro shall return a pointer to the next **cmsghdr** structure, or a null pointer if this structure is the last **cmsghdr** in the ancillary data.

CMSG_FIRSTHDR(*mhdr*)
    If the argument is a pointer to a **msghdr** structure, this macro shall return a pointer to the first **cmsghdr** structure in the ancillary data associated with this **msghdr** structure, or a null pointer if there is no ancillary data associated with the **msghdr** structure.

The *<sys/socket.h>* header shall define the **linger** structure, which shall include at least the following members:

```
int  l_onoff   Indicates whether linger option is enabled.
int  l_linger  Linger time, in seconds.
```

The *<sys/socket.h>* header shall define the following symbolic constants with distinct values:

SOCK_DGRAM
    Datagram socket.

SOCK_RAW
    [RS] ⌦ Raw Protocol Interface. ⌫

SOCK_SEQPACKET
    Sequenced-packet socket.

SOCK_STREAM

Byte-stream socket.

The *<sys/socket.h>* header shall define the following symbolic constant for use as the *level* argument of *setsockopt()* and *getsockopt()*.

SOL_SOCKET
: Options to be accessed at socket level, not protocol level.

The *<sys/socket.h>* header shall define the following symbolic constants with distinct values for use as the *option_name* argument in *getsockopt()* or *setsockopt()* calls (see XSH *Use of Options*):

SO_ACCEPTCONN	Socket is accepting connections.
SO_BROADCAST	Transmission of broadcast messages is supported.
SO_DEBUG	Debugging information is being recorded.
SO_DONTROUTE	Bypass normal routing.
SO_ERROR	Socket error status.
SO_KEEPALIVE	Connections are kept alive with periodic messages.
SO_LINGER	Socket lingers on close.
SO_OOBINLINE	Out-of-band data is transmitted in line.
SO_RCVBUF	Receive buffer size.
SO_RCVLOWAT	Receive ``low water mark''.
SO_RCVTIMEO	Receive timeout.
SO_REUSEADDR	Reuse of local addresses is supported.
SO_SNDBUF	Send buffer size.
SO_SNDLOWAT	Send ``low water mark''.
SO_SNDTIMEO	Send timeout.
SO_TYPE	Socket type.

The *<sys/socket.h>* header shall define the following symbolic constant for use as the maximum *backlog* queue length which may be specified by the *backlog* field of the *listen()* function:

SOMAXCONN
: The maximum *backlog* queue length.

The *<sys/socket.h>* header shall define the following symbolic constants with distinct values for use as the valid values for the *msg_flags* field in the **msghdr** structure, or the *flags* parameter in *recv()*, *recvfrom()*, *recvmsg()*, *send()*, *sendmsg()*, or *sendto()* calls:

MSG_CTRUNC	Control data truncated.
MSG_DONTROUTE	Send without using routing tables.
MSG_EOR	Terminates a record (if supported by the protocol).
MSG_OOB	Out-of-band data.
MSG_NOSIGNAL	No SIGPIPE generated when an attempt to send is made on a stream-oriented socket that is no longer connected.

MSG_PEEK
  Leave received data in queue.
MSG_TRUNC
  Normal data truncated.
MSG_WAITALL
  Attempt to fill the read buffer.

The *<sys/socket.h>* header shall define the following symbolic constants with distinct values:

AF_INET
  Internet domain sockets for use with IPv4 addresses.
AF_INET6
  [IP6] ⌦ Internet domain sockets for use with IPv6 addresses. ⌫
AF_UNIX
  UNIX domain sockets.
AF_UNSPEC
  Unspecified.

The value of AF_UNSPEC shall be 0.

The *<sys/socket.h>* header shall define the following symbolic constants with distinct values:

SHUT_RD
  Disables further receive operations.
SHUT_RDWR
  Disables further send and receive operations.
SHUT_WR
  Disables further send operations.

The *<sys/socket.h>* header shall define the **size_t** and **ssize_t** types as described in *<sys/types.h>*.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
int     accept(int, struct sockaddr *restrict, socklen_t *restrict);
int     bind(int, const struct sockaddr *, socklen_t);
int     connect(int, const struct sockaddr *, socklen_t);
int     getpeername(int, struct sockaddr *restrict, socklen_t *restrict);
int     getsockname(int, struct sockaddr *restrict, socklen_t *restrict);
int     getsockopt(int, int, int, void *restrict, socklen_t *restrict);
int     listen(int, int);
ssize_t recv(int, void *, size_t, int);
ssize_t recvfrom(int, void *restrict, size_t, int,
        struct sockaddr *restrict, socklen_t *restrict);
ssize_t recvmsg(int, struct msghdr *, int);
ssize_t send(int, const void *, size_t, int);
ssize_t sendmsg(int, const struct msghdr *, int);
ssize_t sendto(int, const void *, size_t, int, const struct sockaddr *,
        socklen_t);
int     setsockopt(int, int, int, const void *, socklen_t);
int     shutdown(int, int);
int     sockatmark(int);
int     socket(int, int, int);
int     socketpair(int, int, int, int [2]);
```

Inclusion of *<sys/socket.h>* may also make visible all symbols from *<sys/uio.h>*.

The following sections are informative.

APPLICATION USAGE

To forestall portability problems, it is recommended that applications not use values larger than 2^{31} -1 for the **socklen_t** type.

The **sockaddr_storage** structure solves the problem of declaring storage for automatic variables which is both large enough and aligned enough for storing the socket address data structure of any family. For example, code with a file descriptor and without the context of the address family can pass a pointer to a variable of this type, where a pointer to a socket address structure is expected in calls such as *getpeername()*, and determine the address family by accessing the received content after the call.

The example below illustrates a data structure which aligns on a 64-bit boundary. An implementation-defined field *_ss_align* following *_ss_pad1* is used to force a 64-bit alignment which covers proper alignment good enough for needs of at least **sockaddr_in6** (IPv6) and **sockaddr_in** (IPv4) address data structures. The size of padding field *_ss_pad1* depends on the chosen alignment boundary. The size of padding field *_ss_pad2* depends on the value of overall size chosen for the total size of the structure. This size and alignment are represented in the above example by implementation-defined (not required) constants _SS_MAXSIZE (chosen value 128) and _SS_ALIGNMENT (with chosen value 8). Constants _SS_PAD1SIZE (derived value 6) and _SS_PAD2SIZE (derived value 112) are also for illustration and not required. The implementation-defined definitions and structure field names above start with an <underscore> to denote implementation private name space. Portable code is not expected to access or reference those fields or constants.

```
/*
 *  Desired design of maximum size and alignment.
 */
#define _SS_MAXSIZE 128
    /* Implementation-defined maximum size. */
#define _SS_ALIGNSIZE (sizeof(int64_t))
    /* Implementation-defined desired alignment. */


/*
 *  Definitions used for sockaddr_storage structure paddings design.
 */
#define _SS_PAD1SIZE (_SS_ALIGNSIZE - sizeof(sa_family_t))
#define _SS_PAD2SIZE (_SS_MAXSIZE - (sizeof(sa_family_t)+ \
                      _SS_PAD1SIZE + _SS_ALIGNSIZE))
struct sockaddr_storage {
    sa_family_t  ss_family;  /* Address family. */
/*
 *  Following fields are implementation-defined.
 */
    char _ss_pad1[_SS_PAD1SIZE];
        /* 6-byte pad; this is to make implementation-defined
           pad up to alignment field that follows explicit in
           the data structure. */
    int64_t _ss_align;  /* Field to force desired structure
                           storage alignment. */
    char _ss_pad2[_SS_PAD2SIZE];
        /* 112-byte pad to achieve desired size,
           _SS_MAXSIZE value minus size of ss_family
           __ss_pad1, __ss_align fields is 112. */
};
```

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

<sys/types.h>, <sys/uio.h>

XSH accept, bind, connect, getpeername, getsockname, getsockopt, listen, recv, recvfrom, recvmsg, send, sendmsg, sendto, setsockopt, shutdown, sockatmark, socket, socketpair

CHANGE HISTORY

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

The **restrict** keyword is added to the prototypes for *accept()*, *getpeername()*, *getsockname()*, *getsockopt()*, and *recvfrom()*.

Issue 7

SD5-XBD-ERN-56 is applied, adding a reference to <sys/types.h> for the **ssize_t** type.

SD5-XBD-ERN-62 is applied.

The MSG_NOSIGNAL symbolic constant is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

This reference page is clarified with respect to macros and symbolic constants, and a declaration for the **size_t** type is added.

POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0067 [355] is applied.

POSIX.1-2008, Technical Corrigendum 2, XBD/TC2-2008/0077 [934] is applied.

End of informative text.

return to top of page

[Main Index | XBD | XSH | XCU | XRAT]

<<< Previous Home Next >>>