NAME

socket − Linux socket interface

SYNOPSIS

#include <sys/socket.h>

sockfd **= socket(int** *socket_family***, int** *socket_type***, int** *protocol***);**

DESCRIPTION

This manual page describes the Linux networking socket layer user interface. The BSD compatible sockets are the uniform interface between the user process and the network protocol stacks in the kernel. The protocol modules are grouped into *protocol families* like **AF_INET**, **AF_IPX**, **AF_PACKET** and *socket types* like **SOCK_STREAM** or **SOCK_DGRAM**. See **socket**(2) for more information on families and types.

Socket Layer Functions

These functions are used by the user process to send or receive packets and to do other socket operations. For more information see their respective manual pages.

socket(2) creates a socket, **connect**(2) connects a socket to a remote socket address, the **bind**(2) function binds a socket to a local socket address, **listen**(2) tells the socket that new connections shall be accepted, and **accept**(2) is used to get a new socket with a new incoming connection. **socketpair**(2) returns two connected anonymous sockets (only implemented for a few local families like **AF_UNIX**)

send(2), **sendto**(2), and **sendmsg**(2) send data over a socket, and **recv**(2), **recvfrom**(2), **recvmsg**(2) receive data from a socket. **poll**(2) and **select**(2) wait for arriving data or a readiness to send data. In addition, the standard I/O operations like **write**(2), **writev**(2), **sendfile**(2), **read**(2), and **readv**(2) can be used to read and write data.

getsockname(2) returns the local socket address and **getpeername**(2) returns the remote socket address. **getsockopt**(2) and **setsockopt**(2) are used to set or get socket layer or protocol options. **ioctl**(2) can be used to set or read some other options.

close(2) is used to close a socket. **shutdown**(2) closes parts of a full-duplex socket connection.

Seeking, or calling **pread**(2) or **pwrite**(2) with a non-zero position is not supported on sockets.

It is possible to do non-blocking I/O on sockets by setting the **O_NONBLOCK** flag on a socket file descriptor using **fcntl**(2). Then all operations that would block will (usually) return with **EAGAIN** (operation should be retried later); **connect**(2) will return **EINPROGRESS** error. The user can then wait for various events via **poll**(2) or **select**(2).

I/O events		
Event	Poll flag	Occurrence
Read	POLLIN	New data arrived.
Read	POLLIN	A connection setup has been completed (for connection-oriented sockets)
Read	POLLHUP	A disconnection request has been initiated by the other end.
Read	POLLHUP	A connection is broken (only for connection-oriented protocols). When the socket is written **SIGPIPE** is also sent.
Write	POLLOUT	Socket has enough send buffer space for writing new data.
Read/Write	POLLIN| POLLOUT	An outgoing **connect**(2) finished.
Read/Write	POLLERR	An asynchronous error occurred.
Read/Write	POLLHUP	The other end has shut down one direction.
Exception	POLLPRI	Urgent data arrived. **SIGURG** is sent then.

An alternative to **poll**(2) and **select**(2) is to let the kernel inform the application about events via a **SIGIO** signal. For that the **O_ASYNC** flag must be set on a socket file descriptor via **fcntl**(2) and a valid signal handler for **SIGIO** must be installed via **sigaction**(2). See the *Signals* discussion below.

Socket Options

These socket options can be set by using **setsockopt**(2) and read with **getsockopt**(2) with the socket level set to **SOL_SOCKET** for all sockets:

SO_ACCEPTCONN

Returns a value indicating whether or not this socket has been marked to accept connections with **listen**(2). The value 0 indicates that this is not a listening socket, the value 1 indicates that this is a listening socket. Can only be read with **getsockopt**(2).

SO_BINDTODEVICE

Bind this socket to a particular device like "eth0", as specified in the passed interface name. If the name is an empty string or the option length is zero, the socket device binding is removed. The passed option is a variable-length null-terminated interface name string with the maximum size of **IFNAMSIZ**. If a socket is bound to an interface, only packets received from that particular interface are processed by the socket. Note that this only works for some socket types, particularly **AF_INET** sockets. It is not supported for packet sockets (use normal **bind**(8) there).

SO_BROADCAST

Set or get the broadcast flag. When enabled, datagram sockets receive packets sent to a broadcast address and they are allowed to send packets to a broadcast address. This option has no effect on stream-oriented sockets.

SO_BSDCOMPAT

Enable BSD bug-to-bug compatibility. This is used by the UDP protocol module in Linux 2.0 and 2.2. If enabled ICMP errors received for a UDP socket will not be passed to the user program. In later kernel versions, support for this option has been phased out: Linux 2.4 silently ignores it, and Linux 2.6 generates a kernel warning (printk()) if a program uses this option. Linux 2.0 also enabled BSD bug-to-bug compatibility options (random header changing, skipping of the broadcast flag) for raw sockets with this option, but that was removed in Linux 2.2.

SO_DEBUG

Enable socket debugging. Only allowed for processes with the **CAP_NET_ADMIN** capability or an effective user ID of 0.

SO_ERROR
Get and clear the pending socket error. Only valid as a **getsockopt**(2). Expects an integer.

SO_DONTROUTE
Don't send via a gateway, only send to directly connected hosts. The same effect can be achieved by setting the **MSG_DONTROUTE** flag on a socket **send**(2) operation. Expects an integer boolean flag.

SO_KEEPALIVE
Enable sending of keep-alive messages on connection-oriented sockets. Expects an integer boolean flag.

SO_LINGER
Sets or gets the **SO_LINGER** option. The argument is a *linger* structure.

```
struct linger {
   int l_onoff;    /* linger active */
   int l_linger;   /* how many seconds to linger for */
};
```

When enabled, a **close**(2) or **shutdown**(2) will not return until all queued messages for the socket have been successfully sent or the linger timeout has been reached. Otherwise, the call returns immediately and the closing is done in the background. When the socket is closed as part of **exit**(2), it always lingers in the background.

SO_OOBINLINE
If this option is enabled, out-of-band data is directly placed into the receive data stream. Otherwise out-of-band data is only passed when the **MSG_OOB** flag is set during receiving.

SO_PASSCRED
Enable or disable the receiving of the **SCM_CREDENTIALS** control message. For more information see **unix**(7).

SO_PEERCRED
Return the credentials of the foreign process connected to this socket. This is only possible for connected **AF_UNIX** stream sockets and **AF_UNIX** stream and datagram socket pairs created using **socketpair**(2); see **unix**(7). The returned credentials are those that were in effect at the time of the call to **connect**(2) or **socketpair**(2). Argument is a *ucred* structure. Only valid as a **getsockopt**(2).

SO_PRIORITY
Set the protocol-defined priority for all packets to be sent on this socket. Linux uses this value to order the networking queues: packets with a higher priority may be processed first depending on the selected device queueing discipline. For **ip**(7), this also sets the IP type-of-service (TOS) field for outgoing packets. Setting a priority outside the range 0 to 6 requires the **CAP_NET_ADMIN** capability.

SO_RCVBUF
Sets or gets the maximum socket receive buffer in bytes. The kernel doubles this value (to allow space for bookkeeping overhead) when it is set using **setsockopt**(2), and this doubled value is returned by **getsockopt**(2). The default value is set by the */proc/sys/net/core/rmem_default* file, and the maximum allowed value is set by the */proc/sys/net/core/rmem_max* file. The minimum (doubled) value for this option is 256.

SO_RCVBUFFORCE (since Linux 2.6.14)
Using this socket option, a privileged (**CAP_NET_ADMIN**) process can perform the same task as **SO_RCVBUF**, but the *rmem_max* limit can be overridden.

SO_RCVLOWAT and **SO_SNDLOWAT**
Specify the minimum number of bytes in the buffer until the socket layer will pass the data to the protocol (**SO_SNDLOWAT**) or the user on receiving (**SO_RCVLOWAT**). These two values are

initialized to 1. **SO_SNDLOWAT** is not changeable on Linux (**setsockopt**(2) fails with the error **ENOPROTOOPT**). **SO_RCVLOWAT** is changeable only since Linux 2.4. The **select**(2) and **poll**(2) system calls currently do not respect the **SO_RCVLOWAT** setting on Linux, and mark a socket readable when even a single byte of data is available. A subsequent read from the socket will block until **SO_RCVLOWAT** bytes are available.

SO_RCVTIMEO and **SO_SNDTIMEO**

Specify the receiving or sending timeouts until reporting an error. The argument is a *struct timeval*. If an input or output function blocks for this period of time, and data has been sent or received, the return value of that function will be the amount of data transferred; if no data has been transferred and the timeout has been reached then −1 is returned with *errno* set to **EAGAIN** or **EWOULDBLOCK** just as if the socket was specified to be non-blocking. If the timeout is set to zero (the default) then the operation will never timeout. Timeouts only have effect for system calls that perform socket I/O (e.g., **read**(2), **recvmsg**(2), **send**(2), **sendmsg**(2)); timeouts have no effect for **select**(2), **poll**(2), **epoll_wait**(2), etc.

SO_REUSEADDR

Indicates that the rules used in validating addresses supplied in a **bind**(2) call should allow reuse of local addresses. For **AF_INET** sockets this means that a socket may bind, except when there is an active listening socket bound to the address. When the listening socket is bound to **INADDR_ANY** with a specific port then it is not possible to bind to this port for any local address. Argument is an integer boolean flag.

SO_SNDBUF

Sets or gets the maximum socket send buffer in bytes. The kernel doubles this value (to allow space for bookkeeping overhead) when it is set using **setsockopt**(2), and this doubled value is returned by **getsockopt**(2). The default value is set by the */proc/sys/net/core/wmem_default* file and the maximum allowed value is set by the */proc/sys/net/core/wmem_max* file. The minimum (doubled) value for this option is 2048.

SO_SNDBUFFORCE (since Linux 2.6.14)

Using this socket option, a privileged (**CAP_NET_ADMIN**) process can perform the same task as **SO_SNDBUF**, but the *wmem_max* limit can be overridden.

SO_TIMESTAMP

Enable or disable the receiving of the **SO_TIMESTAMP** control message. The timestamp control message is sent with level **SOL_SOCKET** and the *cmsg_data* field is a *struct timeval* indicating the reception time of the last packet passed to the user in this call. See **cmsg**(3) for details on control messages.

SO_TYPE

Gets the socket type as an integer (like **SOCK_STREAM**). Can only be read with **getsockopt**(2).

Signals

When writing onto a connection-oriented socket that has been shut down (by the local or the remote end) **SIGPIPE** is sent to the writing process and **EPIPE** is returned. The signal is not sent when the write call specified the **MSG_NOSIGNAL** flag.

When requested with the **FIOSETOWN fcntl**(2) or **SIOCSPGRP ioctl**(2), **SIGIO** is sent when an I/O event occurs. It is possible to use **poll**(2) or **select**(2) in the signal handler to find out which socket the event occurred on. An alternative (in Linux 2.2) is to set a real-time signal using the **F_SETSIG fcntl**(2); the handler of the real time signal will be called with the file descriptor in the *si_fd* field of its *siginfo_t*. See **fcntl**(2) for more information.

Under some circumstances (e.g., multiple processes accessing a single socket), the condition that caused the **SIGIO** may have already disappeared when the process reacts to the signal. If this happens, the process should wait again because Linux will resend the signal later.

/proc interfaces

The core socket networking parameters can be accessed via files in the directory */proc/sys/net/core/* .

rmem_default
: contains the default setting in bytes of the socket receive buffer.

rmem_max
: contains the maximum socket receive buffer size in bytes which a user may set by using the **SO_RCVBUF** socket option.

wmem_default
: contains the default setting in bytes of the socket send buffer.

wmem_max
: contains the maximum socket send buffer size in bytes which a user may set by using the **SO_SNDBUF** socket option.

message_cost and *message_burst*
: configure the token bucket filter used to load limit warning messages caused by external network events.

netdev_max_backlog
: Maximum number of packets in the global input queue.

optmem_max
: Maximum length of ancillary data and user control data like the iovecs per socket.

Ioctls

These operations can be accessed using **ioctl**(2):

error = **ioctl(***ip_socket***,** *ioctl_type***,** *&value_result***);**

SIOCGSTAMP
: Return a *struct timeval* with the receive timestamp of the last packet passed to the user. This is useful for accurate round trip time measurements. See **setitimer**(2) for a description of *struct timeval*. This ioctl should only be used if the socket option **SO_TIMESTAMP** is not set on the socket. Otherwise, it returns the timestamp of the last packet that was received while **SO_TIMESTAMP** was not set, or it fails if no such packet has been received, (i.e., **ioctl**(2) returns −1 with *errno* set to **ENOENT**).

SIOCSPGRP
: Set the process or process group to send **SIGIO** or **SIGURG** signals to when an asynchronous I/O operation has finished or urgent data is available. The argument is a pointer to a *pid_t*. If the argument is positive, send the signals to that process. If the argument is negative, send the signals to the process group with the ID of the absolute value of the argument. The process may only choose itself or its own process group to receive signals unless it has the **CAP_KILL** capability or an effective UID of 0.

FIOASYNC
: Change the **O_ASYNC** flag to enable or disable asynchronous I/O mode of the socket. Asynchronous I/O mode means that the **SIGIO** signal or the signal set with **F_SETSIG** is raised when a new I/O event occurs.

  Argument is an integer boolean flag. (This operation is synonymous with the use of **fcntl**(2) to set the **O_ASYNC** flag.)

SIOCGPGRP
: Get the current process or process group that receives **SIGIO** or **SIGURG** signals, or 0 when none is set.

Valid **fcntl**(2) operations:

FIOGETOWN
The same as the **SIOCGPGRP ioctl**(2).

FIOSETOWN
The same as the **SIOCSPGRP ioctl**(2).

VERSIONS

SO_BINDTODEVICE was introduced in Linux 2.0.30. **SO_PASSCRED** is new in Linux 2.2. The */proc* interfaces was introduced in Linux 2.2. **SO_RCVTIMEO** and **SO_SNDTIMEO** are supported since Linux 2.3.41. Earlier, timeouts were fixed to a protocol-specific setting, and could not be read or written.

NOTES

Linux assumes that half of the send/receive buffer is used for internal kernel structures; thus the values in the corresponding */proc* files are twice what can be observed on the wire.

Linux will only allow port re-use with the **SO_REUSEADDR** option when this option was set both in the previous program that performed a **bind**(2) to the port and in the program that wants to re-use the port. This differs from some implementations (e.g., FreeBSD) where only the later program needs to set the **SO_REUSEADDR** option. Typically this difference is invisible, since, for example, a server program is designed to always set this option.

BUGS

The **CONFIG_FILTER** socket options **SO_ATTACH_FILTER** and **SO_DETACH_FILTER** are not documented. The suggested interface to use them is via the libpcap library.

SEE ALSO

getsockopt(2), **setsockopt**(2), **socket**(2), **capabilities**(7), **ddp**(7), **ip**(7), **packet**(7), **tcp**(7), **udp**(7), **unix**(7)

COLOPHON

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.