tcpcp_stop()

NAME

tcpcp_stop - stop the connected TCP socket from sending and receiving packets

SYNOPSIS

#include <tcpcp.h>

int tcpcp_stop(int *fd***, int** *flag***, struct result_list *****reslst***);**

DESCRIPTION

The function **tcpcp_stop**() is an API for TCPCP2.

If the value of the argument *flag* is **TCPCP_EXEC_ONE**, **tcpcp_stop**() stops the connected TCP socket *fd* from sending and receiving packets.

If the value of *flag* is **TCPCP_EXEC_ALL**, **tcpcp_stop**() stops all connected TCP sockets of the calling process from sending and receiving packets.
In this case, you must create a dummy socket *fd* before calling **tcpcp_stop**().

In either case, **tcpcp_stop**() returns a malloc'ed Processing Result List *reslst*.

```
struct result_list {
    unsigned int fd_all;  /* The number of target sockets. */
    unsigned int fd_ng;   /* The number of failed sockets. */
    struct fdr fdrs[0];
};

struct fdr {
    int fd;             /* Socket descriptor. */
    int result;           /* On success, zero is set.
                         * On failure, errno is set. */
};
```

The socket, which has stopped sending and receiving packets, can resume sending and receiving them by calling the function **tcpcp_start**().

RETURN VALUE

If **tcpcp_stop**() succeeds, **TCPCP_RET_OK** is returned.
Otherwise, **TCPCP_RET_NG** is returned and error number *errno* is set appropriately.

ERRORS

EINVAL
The argument *flag* is neither **TCPCP_EXEC_ONE** nor **TCPCP_EXEC_ALL**.

ENOSYS
The system cannot link a kernel module of TCPCP2.

EBADF
The argument *fd* is an invalid descriptor.

ENOTSOCK
The argument *fd* is a file, not a socket.

EPFNOSUPPORT
The socket family is neither IPv4 nor IPv6.

ESOCKTNOSUPPORT
The socket type is not byte stream.

EPROTONOSUPPORT
The protocol type of the socket is not TCP.

EBADFD
The socket is not connected.

EBUSY
The socket is currently locked.

EFAULT
The argument *reslst* is an invalid pointer.

ENOMEM
Out of memory.

tcpcp_get_si()

- NAME

tcpcp_get_si - retrieve the TCP socket information (TCP-SI)

- SYNOPSIS

#include <tcpcp.h>

int tcpcp_get_si(int *fd*, **int** *flag*, **void *****bufptr*, **unsigned int ****size*, **struct result_list *****reslst*);

- DESCRIPTION

The function **tcpcp_get_si**() is an API for TCPCP2.

If the value of the argument *flag* is **TCPCP_EXEC_ONE**, you can retrieve the TCP-SI of the connected TCP socket *fd*.

If the value of *flag* is **TCPCP_EXEC_ALL**, you can retrieve the TCP-SIs of all connected TCP sockets of a calling process.
In this case, you must create a dummy socket *fd* before calling **tcpcp_get_si**().

In either case, **tcpcp_get_si**() returns the following values:

- malloc'ed TCP-SI *bufptr*
- size of TCP-SI
- malloc'ed Processing Result List *reslst*

```
struct result_list {
    unsigned int fd_all; /* The number of target sockets. */
    unsigned int fd_ng;  /* The number of failed sockets. */
    struct fdr fdrs[0];
};

struct fdr {
    int fd;             /* Socket descriptor. */
    int result;          /* On success, zero is set.
                        * On failure, errno is set. */
};
```

TCP-SI contains the following information:

- connection identifier
    - IP version (4 or 6)
    - source/destination IP address
    - source/destination TCP port number

- sequence number
    - sequence number of next new byte to send
    - sequence number of next new byte expected
    - window received from peer
    - window advertised to peer
    - send window scale
    - receive window scale

- timestamp
    - cached timestamp from peer
    - current locally generated timestamp

- options on socket

    - socket level
        - protocol-defined priority for all packets to be sent
        - size of socket send buffer in bytes
        - size of socket receive buffer in bytes
        - how long it should linger

    - IPv4 level
        - unicast TTL
        - Path MTU Discovery setting
        - TOS
        - IP options to be sent

    - IPv6 level
        - unicast hop limit
        - Path MTU Discovery setting

    - TCP level
        - state of Nagle algorithm
        - number of allowed keep-alive probes
        - time before keep-alive takes place
        - time interval between keep-alive probes

- TCP packets in socket send buffer and socket receive buffer
    - flags of TCP header
    - sequence number of first byte
    - data

RETURN VALUE

If **tcpcp_get_si**() succeeds, **TCPCP_RET_OK** is returned.
Otherwise, **TCPCP_RET_NG** is returned and error number *errno* is set appropriately.

ERRORS

EINVAL
The argument *flag* is neither **TCPCP_EXEC_ONE** nor **TCPCP_EXEC_ALL**.

ENOSYS
The system cannot link a kernel module of TCPCP2.

EBADF
The argument *fd* is an invalid descriptor.

ENOTSOCK
The argument *fd* is a file, not a socket.

EPFNOSUPPORT
The socket family is neither IPv4 nor IPv6.

ESOCKTNOSUPPORT
The socket type is not byte stream.

EPROTONOSUPPORT
The protocol type of the socket is not TCP.

EBADFD
The socket is not connected.

EBUSY
The socket is currently locked.

EFAULT
Either argument *bufptr* or *reslst* is an invalid pointer.

ENOMEM
Out of memory.

tcpcp_set_si()

NAME

tcpcp_set_si - set the TCP socket information (TCP-SI) in an unconnected TCP socket

SYNOPSIS

#include <tcpcp.h>

int tcpcp_set_si(int *fd*, **int** *flag*, **const void** ***bufptr*, **unsigned int** **size*, **struct result_list** ***reslst*);

DESCRIPTION

The function **tcpcp_set_si**() is an API for TCPCP2.

If the value of the argument *flag* is **TCPCP_EXEC_ONE**, you can set the TCP-SI, which has retrieved by calling the function **tcpcp_get_si**() with the value **TCPCP_EXEC_ONE** of *flag*, in the unconnected TCP socket *fd*.
In this case, you must create *fd* before calling **tcpcp_set_si**().

If the value of *flag* is **TCPCP_EXEC_ALL**, you can create unconnected TCP sockets and set their TCP-SIs, which were retrieved by calling **tcpcp_get_si**() with the value **TCPCP_EXEC_ALL** of *flag*, in these sockets.
In this case, you must create a dummy socket *fd* before calling **tcpcp_set_si**().

In either case, you must set TCP-SI *bufptr* and the size of it.

tcpcp_set_si() returns a malloc'ed Processing Result List *reslst*.

```
struct result_list {
    unsigned int fd_all; /* The number of target sockets. */
    unsigned int fd_ng;  /* The number of failed sockets. */
    struct fdr fdrs[0];
};

struct fdr {
    int fd;             /* Old socket descriptor. */
    int result;          /* On success, a new socket descriptor is set.
                        * On failure, errno is set. */
};
```

RETURN VALUE

If **tcpcp_set_si**(), **TCPCP_RET_OK** is returned.
Otherwise, **TCPCP_RET_NG** is returned and error number *errno* is set appropriately.

ERRORS

EINVAL
The argument *flag* is neither **TCPCP_EXEC_ONE** nor **TCPCP_EXEC_ALL**.

ENOSYS
The system cannot link a kernel module of TCPCP2.

EBADF
The argument *fd* is an invalid descriptor.

ENOTSOCK
The argument *fd* is a file, not a socket.

EPFNOSUPPORT
The socket family is neither IPv4 nor IPv6.

ESOCKTNOSUPPORT
The socket type is not byte stream.

EPROTONOSUPPORT
The protocol type of the socket is not TCP.

EBADFD
The socket is connected.

EBUSY
The socket is currently locked.

EFAULT
Either argument *bufptr* or *reslst* is an invalid pointer.

ENOMEM
Out of memory.

tcpcp_start()

NAME

tcpcp_start - allow the TCP socket to resume sending and receiving packets

SYNOPSIS

#include <tcpcp.h>

int tcpcp_start(int *fd***, int** *flag***, struct result_list *****reslst***);**

DESCRIPTION

The function **tcpcp_start**() is an API for TCPCP2.

If the value of the argument *flag* is **TCPCP_EXEC_ONE**, then the TCP socket *fd*, which has stopped sending and receiving packets because either function **tcpcp_set_si**() or **tcpcp_stop**() was called, can resume sending and receiving them.

If the value of *flag* is **TCPCP_EXEC_ALL**, all TCP sockets of the calling process can resume sending and receiving packets.
In this case, you must create a dummy socket *fd* before calling **tcpcp_start**().

In either case, **tcpcp_start**() returns a malloc'ed Processing Result List *reslst*.

```
struct result_list {
    unsigned int fd_all;  /* The number of target sockets. */
    unsigned int fd_ng;   /* The number of failed sockets. */
    struct fdr fdrs[0];
};

struct fdr {
    int fd;             /* Socket descriptor. */
    int result;          /* On success, zero is set.
                        * On failure, errno is set. */
};
```

RETURN VALUE

If **tcpcp_start**(), **TCPCP_RET_OK** is returned.
Otherwise, **TCPCP_RET_NG** is returned and error number *errno* is set appropriately.

ERRORS

EINVAL
The argument *flag* is neither **TCPCP_EXEC_ONE** nor **TCPCP_EXEC_ALL**.

ENOSYS
The system cannot link a kernel module of TCPCP2.

EBADF
The argument *fd* is an invalid descriptor.

ENOTSOCK
The argument *fd* is a file, not a socket.

EPFNOSUPPORT
The socket family is neither IPv4 nor IPv6.

ESOCKTNOSUPPORT
The socket type is not byte stream.

EPROTONOSUPPORT
The protocol type of the socket is not TCP.

EBUSY
The socket is currently locked.

ENODEV
You try call **tcpcp_start**() for the TCP/IPv6 socket, for which has be set the TCP-SI containing a link-local address by calling **tcpcp_set_si**(), but the device that the socket attempts to use does not exist.

EFAULT
The argument *reslst* is an invalid pointer.

ENOMEM
Out of memory.