

# NORTi Socket Interface ユーザーズガイド

2022年03月版



株式会社 ミスポ

## 目次

|                           |    |
|---------------------------|----|
| 第1章 導入.....               | 1  |
| 1.1 はじめに.....             | 1  |
| 1.2 ファイル構成 .....          | 1  |
| 1.3 ソケットとは .....          | 1  |
| 1.4 BSD ソケットとの比較.....     | 2  |
| 1.5 制限事項.....             | 2  |
| 第2章 ソケットの動作と共通定義 .....    | 3  |
| 2.1 ソケットの状態.....          | 3  |
| 2.2 内部で使用する OS リソース ..... | 4  |
| 2.3 コンフィグレーション .....      | 4  |
| 2.4 エラーコード .....          | 4  |
| 第3章 関数定義.....             | 5  |
| 3.1 関数一覧.....             | 5  |
| sock_ini.....             | 6  |
| socket.....               | 7  |
| bind.....                 | 8  |
| listen.....               | 10 |
| accept.....               | 11 |
| connect .....             | 12 |
| send.....                 | 13 |
| recv.....                 | 14 |
| sendto .....              | 15 |
| recvfrom.....             | 16 |
| close.....                | 17 |
| shutdown.....             | 18 |
| setsockopt .....          | 19 |
| getsockopt .....          | 20 |
| nsl_get_err .....         | 21 |
| inet_addr .....           | 22 |
| inet_ntoa.....            | 23 |
| inet_aton.....            | 24 |
| inet_pton.....            | 25 |
| inet_ntop.....            | 26 |
| gethostbyname.....        | 27 |

## NORTi Socket Interface

|                   |    |
|-------------------|----|
| getpeername ..... | 28 |
| getsockname ..... | 29 |
| getaddrinfo ..... | 30 |
| freeaddrinfo..... | 32 |
| select .....      | 33 |
| ioctl .....       | 34 |

---

## 第 1 章 導入

---

### 1.1 はじめに

NORTi Socket Interface（以降ソケットインターフェース）を使用すると NORTi プロトコルスタックで BSD ソケットインターフェースを使用することができます。BSD ソケットを使用して記述されたアプリケーションプログラムを NORTi に移植することができます。

### 1.2 ファイル構成

ソケットインターフェースは次のファイルで構成されています。

#### **nonsock.h          ソケットインターフェースヘッダファイル**

ソケットインターフェースを使用する全てのファイルで `#include` する必要があります。このヘッダファイルにはソケットインターフェースで使用する全ての構造体と関数のプロトタイプ宣言が含まれています。

#### **nonsock.c          ソケットインターフェースソースファイル**

ソケットインターフェースのソースファイルです。ソケットインターフェースを使用する為にはこのファイルと `nonedns.c`、`nondhcp.c`、`n4dxxx.xxx` または `n4nxxx.xxx` をアプリケーションにリンクしてください。

### 1.3 ソケットとは

ソケットはネットワーク通信を行うためのエンドポイントを提供する抽象的なものです。アプリケーションプログラムはソケットを生成する場合、ソケットインターフェースに要求します。`socket` 関数は新たに生成されたソケットの値を `int` 型の値を返します。アプリケーションは特定のアドレスに固定されずにソケットを作成することができ、ソケットを使用するたびに終点アドレスを決めることができます（UDP エンドポイント）。あるいはあらかじめ終点アドレスを固定化して使用することもできます（TCP エンドポイント）

## 1.4 BSD ソケットとの比較

ソケットインターフェースは BSD ソケットインターフェースを基にインプリメントされています。しかし、これら 2 つにはいくつかの違いがあります。以下は相違内容です。

| 関数                                 | NORTi Socket Interface                                                                                                                                             | BSD Socket Interface                                                                                                                                                                             |
|------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sock_init                          | NORTi 独自                                                                                                                                                           |                                                                                                                                                                                                  |
| listen                             | パラメータ backlog は未使用です。<br>キューイングの機能は未サポート                                                                                                                           | カーネルがキューイングする接続の最大数を指定するために使用される                                                                                                                                                                 |
| send<br>recv<br>sendto<br>recvfrom | パラメータ flag は未使用                                                                                                                                                    | MSG オプションをカーネルに指定する                                                                                                                                                                              |
| shutdown                           | パラメータ howto は SHUT_RDWR のみサポート                                                                                                                                     | SHUT_RD、SHUT_WD、SHUT_RDWR を指定可能                                                                                                                                                                  |
| setsockopt<br>getsockopt           | 以下の機能のみサポート<br>SO_SOCKET<br>SO_BROADCAST<br>SO_RCVBUF<br>SO_SNDBUF<br>SO_RCVTIMEO, SO_SNDTIMEO<br>SO_CLSTIMEO (NORTi 独自)<br>IP_TOS, IP_TTL                         | 他に多くのオプションをサポート                                                                                                                                                                                  |
| ns_l_get_err                       | NORTi 独自                                                                                                                                                           |                                                                                                                                                                                                  |
| select                             | <ul style="list-style-type: none"> <li>例外状態検出を未サポート</li> <li>確立済み接続の検出を未サポート</li> <li>読み出し(受信)において FIN または RST を受信した場合のみ検出可能</li> <li>帯域外のデータチェックを未サポート</li> </ul> | <ul style="list-style-type: none"> <li>例外状態を検出可能</li> <li>listen による確立待ち/確立済み接続のキューイングで確立済み接続がキューに入った場合を検出可能</li> <li>読み出し(受信)/書き込み(送信)において接続がクローズされた場合を検出可能</li> <li>帯域外データチェックを検出可能</li> </ul> |
| ioctl                              | FIONREAD リクエストのみサポート                                                                                                                                               | 他に多くのリクエストをサポート                                                                                                                                                                                  |

## 1.5 制限事項

NORTi Version4 以外での動作は保証対象外です。

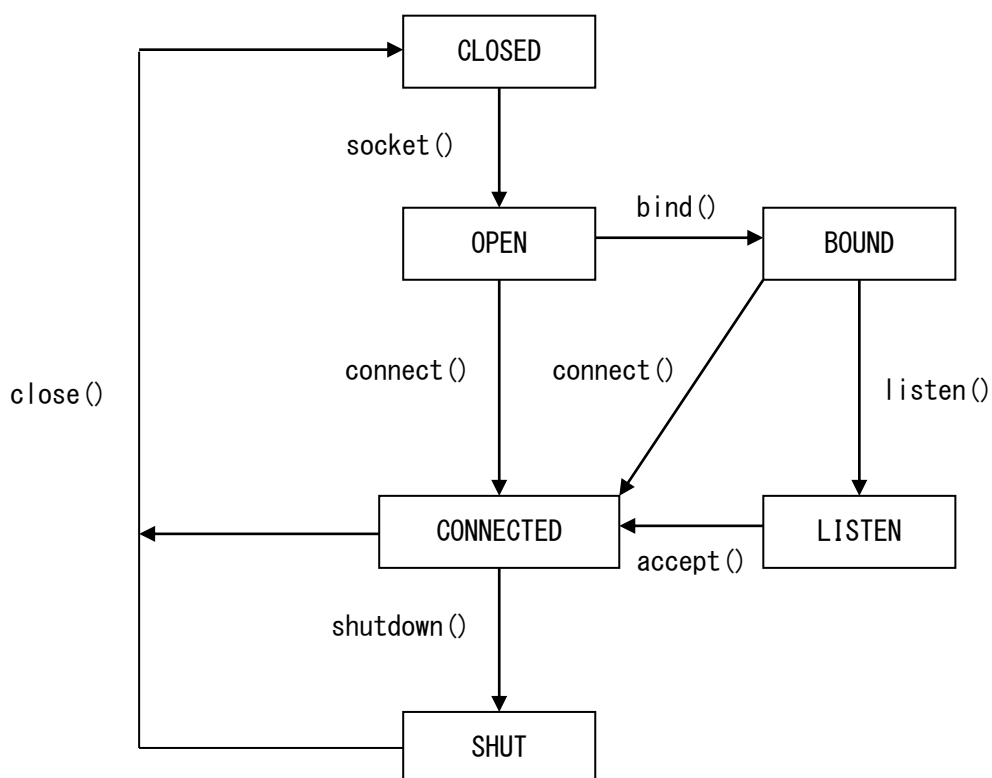
---

## 第 2 章 ソケットの動作と共通定義

---

### 2.1 ソケットの状態

以下のダイアグラムは、関数呼び出しによって変化するソケットの状態を表しています。



## 2.2 内部で使用する OS リソース

次のリソースが使用されます。各リソースの ID は内部で自動的に設定されます。

|           |                                                                                                                                              |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------|
| TCP 受付口   | 受動接続を行う毎に 1 つ<br>listen を呼び出す毎に 1 つ生成します。                                                                                                    |
| TCP 通信端点  | ソケット毎に 1 つ<br>accept または connect で生成され close で削除されます。<br>各通信端点は次のバッファサイズがデフォルトで使用されます。<br>TCP 送信バッファのサイズ 8192byte<br>TCP 受信バッファのサイズ 8192byte |
| UDP 通信端点  | ソケット毎に 1 つ                                                                                                                                   |
| 可変長メモリプール | 1 つ<br>$\text{MAX\_SOCK} \times (\text{RCV\_BUF\_SIZE} + \text{SEND\_BUF\_SIZE})$<br>デフォルトで 32Kbyte 使用します。                                   |

※必要に応じて受付口、通信端点、可変長メモリプール ID の最大値を設定してください。

## 2.3 コンフィグレーション

使用するソケットの数、送受信バッファのサイズを変更できます。特に必要がなければデフォルトのまま使用してください。

次の定義は nonsock.c をコンパイルする際にマクロ指定することで値を変更できます。

( ) 内はデフォルト値です。

|                 |                              |
|-----------------|------------------------------|
| MAX_SOCKET      | 使用するソケットの最大数 (8)             |
| SEND_BUF_SIZE   | TCP 通信で使用する送信バッファのサイズ (8192) |
| RCV_BUF_SIZE    | TCP 通信で使用する受信バッファのサイズ (8192) |
| MAX_SELECT_INFO | select 管理テーブルのエントリ数 (8)      |

例) shc <op> -def= MAX\_SOCKET =20 nonsock.c      SHC の場合

MAX\_SELECT\_INFO には、select() で監視するソケットの最大数を定義してください。

## 2.4 エラーコード

全ての関数はエラーの場合 -1 でリターンします。nsl\_get\_err 関数を呼び出すことでエラーの詳細を獲得できます。

---

## 第 3 章 関数定義

---

### 3.1 関数一覧

|               |                                          |
|---------------|------------------------------------------|
| sock_init     | ソケットインターフェースの初期化                         |
| socket        | ソケットの生成                                  |
| bind          | 指定されたローカルアドレスとの関連付けを行う                   |
| listen        | コネクションの受付を開始する                           |
| accept        | コネクション要求を受け付ける                           |
| connect       | コネクションを確立する                              |
| send          | データ送信                                    |
| recv          | データ受信                                    |
| sendto        | 指定した宛先にデータを送信                            |
| recvfrom      | データを受信し、始点アドレスを格納                        |
| close         | ソケットを閉じる                                 |
| shutdown      | send および recv の呼び出し、またはどちらか一方の呼び出しを無効にする |
| setsockopt    | ソケットオプションを設定する                           |
| getsockopt    | ソケットオプションを取得する                           |
| ns_l_get_err  | ソケット関数のエラーの詳細を取り出す                       |
| inet_addr     | IPv4 のアドレス文字列を 32bit バイナリアドレスへ変換する       |
| inet_ntoa     | 32bit バイナリアドレスをドット付き 10 進文字列に変換する        |
| inet_aton     | IPv4 のアドレス文字列を 32bit バイナリアドレスへ変換する       |
| inet_pton     | IPv4 および IPv6 のアドレス文字列をバイナリアドレスへ変換する     |
| inet_ntop     | ネットワークアドレス構造体で表しているバイナリアドレスを文字列へ変換する     |
| gethostbyname | ホスト名からホスト情報を検索する                         |
| getpeername   | ソケットのリモートプロトコルアドレスを返す                    |
| getsockname   | ソケットのローカルプロトコルアドレスを返す                    |
| getaddrinfo   | ホスト名とポート番号に該当するネットワークアドレス構造体を返す          |
| freeaddrinfo  | getaddrinfo 関数による割り当てられたリソースを解放する        |
| select        | 入出力の多重化                                  |
| ioctl         | ソケットの制御                                  |

---

## sock\_ini

---

【機能】 ソケットインターフェースの初期化

【形式】 `int sock_ini(void)`

【戻値】   0       正常終了  
         -1       エラー

【解説】 ソケットインターフェースの初期化を行います。ソケットインターフェースの各関数を使用する前に一度だけ呼び出してください。

エラーの場合 `nsi_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード   | 値    | 内容               |
|----------|------|------------------|
| E_MEMORY | 1017 | 可変長メモリプールが生成できない |

---

## socket

---

【機能】 ソケットの生成

【形式】 `int socket(int af, int type, int protocol)`

af        アドレスファミリ

type     プロトコルタイプ

protocol 未使用(0 を指定)

【戻値】 0 以上    正常終了(ソケットディスクリプタを返します)

-1        エラー

【解説】 ソケットを生成します。

af は IPv4 を使用する場合は AF\_INET を、IPv6 を使用する場合は AF\_INET6 を指定します。

type は TCP を使用する場合は SOCK\_STREAM を、UDP を使用する場合は SOCK\_DGRAM を指定します。

エラーの場合 `ns1_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード            | 値    | 内容                                    |
|-------------------|------|---------------------------------------|
| E_SOCKETNOTINIT   | 1001 | <code>sock_ini()</code> で初期化が行われていません |
| E_AFNOSUPPORT     | 1002 | 指定されたアドレスファミリがサポートされていません             |
| E_SOCKETNOSUPPORT | 1003 | 指定されたソケットタイプはこのアドレスファミリでサポートされていません   |
| E_PROTONOSUPPORT  | 1004 | 指定されたプロトコルタイプはサポートされていません             |
| E_MFILE           | 1005 | これ以上のソケットディスクリプタが利用できません              |
| E_INVALID         | 1008 | プロトコルスタックは未初期化                        |

---

## bind

---

【機能】 指定されたローカルアドレスとの関連付けを行う

【形式】 `int bind(int sockfd, const struct sockaddr *myaddr, socklen_t addrlen)`

`sockfd` ソケットディスクリプタ

`myaddr` ローカルアドレスが格納されている構造体へのポインタ

`addrlen` アドレスが格納されている構造体のサイズ

【戻値】 0 以上 正常終了

-1 エラー

【解説】 `bind` 関数は `connect` 関数、`listen` 関数を使用する前に使用します。

コネクション型 (ストリーム) ソケット、コネクションレス型 (データグラム) ソケットのいずれかに使用します。この関数を呼び出すとメモリ資源を確保し `myaddr` で指定したローカルアドレスをソケットに割り当てます。インターネット・アドレス・ファミリは次の3つで構成されます。

1) アドレスファミリ

2) ホストアドレス

3) アプリケーションを識別するためのポート番号

アプリケーションが `bind` 関数を呼び出したとき、`sockaddr_in` は `sockaddr` にキャストしパラメータとして渡されます。

```
struct sockaddr_in {
    sa_family_t    sin_family;           インターネットファミリ
    in_port_t      sin_port;             ポート番号
    struct in_addr sin_addr;              ホストのインターネットアドレス
    char           sin_zero[8];          予約
};

struct in_addr {
    union {
        struct { uint8_t s_b1, s_b2, s_b3, s_b4; } _S_un_b;
        struct { uint16_t s_w1, s_w2; } _S_un_w;
        uint32_t _S_addr;
    } _S_un;
};
```

```
#define s_addr _S_un._S_addr
};
```

エラーの場合 `ns_l_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード         | 値    | 内容                             |
|----------------|------|--------------------------------|
| E_SOCKNOTINIT  | 1001 | sock_ini() で初期化が行われていません       |
| E_AFNOSUPPORT  | 1002 | 指定されたアドレスファミリがサポートされていません      |
| E_ADDRINUSE    | 1006 | 指定されたアドレスは既に使用されています           |
| E_NOTSOCK      | 1007 | ディスクリプタが正しくありません               |
| E_INVAL        | 1008 | ソケットはバインディングの準備ができていません        |
| E_ADDRNOTAVAIL | 1009 | 指定されたアドレスが正しくありません             |
| E_FAULT        | 1016 | addr len が不正または myaddr が NULL  |
| E_NETDOWN      | 1018 | ID 不足により通信端点の生成に失敗かオプションの設定に失敗 |

【例】

```
struct sockaddr_in addr;
addr.sin_family = AF_INET;
addr.sin_port = htons(100);
addr.sin_addr.s_addr = inet_addr( "192.168.0.1" );
err = bind(s, (struct sockaddr *)&addr, sizeof(addr));
```

---

## listen

---

【機能】 コネクションの受付を開始する

【形式】 `int listen(int sockfd, int backlog)`  
           sockfd ソケットディスクリプタ  
           backlog コネクションキューの最大長 (1 固定)

【戻値】   0       正常終了  
          -1       エラー

【解説】 ローカルアドレスにバインドされたソケットでコネクションの受付を開始します。  
 この関数は通常 `socket` 関数および `bind` 関数の実行後、`accept` 関数の前に呼ぶ必要があります。この関数はTCPのサーバーのみが使用する関数です。パラメータ `backlog` は現バージョンでは1固定になります。  
 エラーの場合 `ns_l_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード        | 値    | 内容                                    |
|---------------|------|---------------------------------------|
| E_SOCKNOTINIT | 1001 | <code>sock_ini()</code> で初期化が行われていません |
| E_ADDRINUSE   | 1006 | 指定されたアドレスは既に使用されています                  |
| E_NOTSOCK     | 1007 | ディスクリプタが正しくありません                      |
| E_INVALID     | 1008 | ローカルアドレスにバインドされていません                  |
| E_ISCONN      | 1011 | ソケットは既に接続されています                       |
| E_OPNOTSUPP   | 1012 | ソケットがコネクション型ではありません                   |
| E_NETDOWN     | 1018 | 受付口の生成に失敗                             |

---

## accept

---

【機能】 コネクション要求を受け付ける

【形式】 `int accept(int sockfd, struct sockaddr *dstaddr, socklen_t *addrlen)`

`sockfd` ソケットディスクリプタ

`dstaddr` 接続されたアドレスを格納するポインタ

`addrlen` アドレスを格納する構造体のサイズ

【戻値】 正の値 正常終了（新しいソケットディスクリプタ）

-1 エラー

戻り値には新たに生成されたクライアントとの TCP コネクションを参照しているディスクリプタが設定されます。

【解説】 `accept` は接続待ちキュー（このバージョンでは 1 つのみキューイング可能）より接続要求を取り出します。その後、新しいソケットを作成し、そのソケットディスクリプタを返します。新しく作成されたソケットは実際の接続を扱います。

`dstaddr` には接続してきた相手側のアドレスが設定されます。

`addrlen` は `dstaddr` のサイズを指定します。

エラーの場合 `ns_l_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード          | 値    | 内容                                                                         |
|-----------------|------|----------------------------------------------------------------------------|
| E_SOCKETNOTINIT | 1001 | <code>sock_init()</code> で初期化が行われていません                                     |
| E_ADDRINUSE     | 1006 | ソケットが使用中                                                                   |
| E_NOTSOCK       | 1007 | ディスクリプタが正しくありません                                                           |
| E_INVALID       | 1008 | <code>accept</code> の前に <code>listen</code> が呼ばれていません                      |
| E_OPNOTSUPP     | 1012 | コネクション型ソケットではありません                                                         |
| E_FAULT         | 1016 | <code>addrlen</code> パラメータが小さすぎるか <code>dstaddr</code> パラメータが NULL を指しています |
| E_MEMORY        | 1017 | 内部でメモリが取得できません                                                             |
| E_NETDOWN       | 1018 | ID 不足により通信端点の生成に失敗か内部エラー                                                   |

---

## connect

---

【機能】 コネクションを確立する

【形式】 `int connect(int sockfd, const struct sockaddr *dstaddr, socklen_t addrlen)`

`sockfd` ソケットディスクリプタ

`dstaddr` 接続するアドレスが格納されたポインタ

`addrlen` アドレスを格納する構造体のサイズ

【戻値】 0 正常終了

-1 エラー

【解説】 `connect` 関数は指定したアドレスへの接続を確立します。呼び出しが成功すると、ソケットはデータの送受信が可能になります。

エラーの場合 `ns_l_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード          | 値    | 内容                                                                                      |
|-----------------|------|-----------------------------------------------------------------------------------------|
| E_SOCKETNOTINIT | 1001 | <code>sock_ini()</code> で初期化が行われていません                                                   |
| E_AFNOSUPPORT   | 1002 | 指定されたファミリーのアドレスは、このソケットで使用できません                                                         |
| E_ADDRINUSE     | 1006 | ソケットのローカルアドレスがすでに用いられているか、指定したポート番号での接続を拒否されました                                         |
| E_NOTSOCK       | 1007 | ディスクリプタが正しくありません                                                                        |
| E_INVALID       | 1008 | ソケットが未生成.                                                                               |
| E_ADDRNOTAVAIL  | 1009 | <code>dstaddr</code> が適当ではありません                                                         |
| E_CONNREFUSED   | 1010 | リモートホストに接続を拒否された                                                                        |
| E_ISCONN        | 1011 | ソケットは既に接続されています                                                                         |
| E_OPNOTSUPP     | 1012 | 未対応のソケットタイプ                                                                             |
| E_FAULT         | 1016 | <code>addrlen</code> パラメータが小さすぎるか <code>dstaddr</code> パラメータが <code>NULL</code> を指しています |
| E_MEMORY        | 1017 | 内部でメモリが取得できません                                                                          |
| E_NETDOWN       | 1018 | ID 不足により通信端点の生成に失敗かオプションの設定に失敗か内部エラー                                                    |

---

## send

---

【機能】 データ送信

【形式】 `int send(int sockfd, const void *buf, size_t len, int flags)`

`sockfd` ソケットディスクリプタ

`buf` 送信データが含まれているバッファへのポインタ

`len` 送信するサイズ

`flags` フラグ（未使用 0）

【戻値】 正の値 送信されたサイズ

-1 エラー

【解説】 接続されているソケットへデータを送信します。成功した場合、実際にソケットに送信されたサイズが返りますが、`len` で指定されたサイズが必ず送信できるとは限りません。

エラーの場合 `ns_l_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード                       | 値    | 内容                                    |
|------------------------------|------|---------------------------------------|
| <code>E_SOCKETNOTINIT</code> | 1001 | <code>sock_ini()</code> で初期化が行われていません |
| <code>E_NOTSOCK</code>       | 1007 | ディスクリプタが正しくありません                      |
| <code>E_INVALID</code>       | 1008 | パラメータ <code>flags</code> が不正です        |
| <code>E_OPNOTSUPP</code>     | 1012 | 未対応のソケットタイプ                           |
| <code>E_NOTCONN</code>       | 1014 | ソケットが未接続です                            |
| <code>E_FAULT</code>         | 1016 | 送信バッファのアドレスが不正です                      |
| <code>E_NETDOWN</code>       | 1018 | 内部エラー                                 |
| <code>E_CONNABRTD</code>     | 1020 | ソケットは切断されました                          |
| <code>E_CONNRESET</code>     | 1021 | リセットによりソケットは切断されました                   |
| <code>E_HOSTUNREACH</code>   | 1022 | 送信がキャンセルされました                         |
| <code>E_TIMEOUT</code>       | 1023 | 送信がタイムアウトしました                         |
| <code>E_ACCESS</code>        | 1025 | ブロードキャストの送信が許可されていません                 |

---

## recv

---

【機能】 データ受信

【形式】 `int recv(int sockfd, void * buf, size_t len, int flags)`

`sockfd` ソケットディスクリプタ

`buf` データを受信するバッファへのポインタ

`len` バッファサイズ

`flags` 0 (未使用)

【戻値】 正の値 受信したサイズ

-1 エラー

【解説】 接続されているソケットからデータを受信します。成功した場合、ソケットから受信したサイズが返ります。

エラーの場合 `ns_l_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード          | 値    | 内容                                    |
|-----------------|------|---------------------------------------|
| E_SOCKETNOTINIT | 1001 | <code>sock_ini()</code> で初期化が行われていません |
| E_NOTSOCK       | 1007 | ディスクリプタが正しくありません                      |
| E_INVALID       | 1008 | パラメータ <code>flags</code> が不正です        |
| E_OPNOTSUPP     | 1012 | 未対応のプロトコルタイプが設定されています                 |
| E_NOTCONN       | 1014 | ソケットが未接続です                            |
| E_FAULT         | 1016 | 受信バッファのアドレスが不正です                      |
| E_NETDOWN       | 1018 | 内部エラー                                 |
| E_CONNRESET     | 1021 | リセットによりソケットは切断されました                   |
| E_TIMEOUT       | 1023 | 送信がタイムアウトしました                         |

---

## sendto

---

【機能】 指定した宛先にデータを送信

【形式】 `int sendto(int sockfd, const void *msg, size_t len, int flags, const struct sockaddr *to, socklen_t tolen)`

`sockfd` ソケットディスクリプタ

`msg` 送信データが含まれているバッファへのポインタ

`len` 送信するサイズ

`flags` フラグ（未使用 0）

`to` 送信するアドレスが格納されたポインタ

`tolen` アドレスを格納する構造体のサイズ

【戻値】 正の値 送信されたサイズ

-1 エラー

【解説】 送信先のアドレスを指定してソケットにデータを送信します。sendto は通常、非コネクション型の UDP（データグラム）通信で使用されます。成功した場合、ソケットに送信されたサイズが返ります。

エラーの場合 `ns_l_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード          | 値    | 内容                                                            |
|-----------------|------|---------------------------------------------------------------|
| E_SOCKETNOTINIT | 1001 | <code>sock_ini()</code> で初期化が行われていません                         |
| E_AFNOSUPPORT   | 1002 | 指定されたファミリーのアドレスはこのソケットで使用できません                                |
| E_ADDRINUSE     | 1006 | ID 不足により通信端点の生成かオプションの設定に失敗                                   |
| E_NOTSOCK       | 1007 | ディスクリプタが正しくありません                                              |
| E_INVALID       | 1008 | パラメータ <code>flags</code> が不正です                                |
| E_ADDRNOTAVAIL  | 1009 | 送信するアドレスが正しくありません                                             |
| E_OPNOTSUPP     | 1012 | 未対応のプロトコルタイプが設定されています                                         |
| E_SHUTDOWN      | 1013 | ソケットは shutdown されています                                         |
| E_FAULT         | 1016 | データバッファが不正、 <code>tolen</code> が小さすぎる、 <code>to</code> が NULL |
| E_NETDOWN       | 1018 | 内部エラー                                                         |
| E_TIMEDOUT      | 1023 | 送信がタイムアウトしました                                                 |
| E_ACCESS        | 1025 | ブロードキャストの送信が許可されていません                                         |

---

## recvfrom

---

【機能】 データを受信し、始点アドレスを格納

【形式】 `int recvfrom(int sockfd, void *msg, size_t len, int flags, struct sockaddr *from, socklen_t *fromlen)`

`sockfd` ソケットディスクリプタ

`msg` 受信するデータを格納するバッファへのポインタ

`len` 受信バッファのサイズ

`flags` フラグ（未使用 0）

`from` 送信側のアドレスを格納するポインタ

`fromlen` アドレスを格納する構造体のサイズ

【戻値】 正の値 受信したサイズ

-1 エラー

【解説】 データを受信し送信側のアドレスを `from` に格納します。`recvfrom` は通常、非コネクション型の UDP（データグラム）通信で使用されます。成功した場合、ソケットから受信したサイズが返ります。

エラーの場合 `ns1_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード                       | 値    | 内容                                                            |
|------------------------------|------|---------------------------------------------------------------|
| <code>E_SOCKETNOTINIT</code> | 1001 | <code>sock_init()</code> で初期化が行われていません                        |
| <code>E_ADDRINUSE</code>     | 1006 | ID 不足により通信端点の生成に失敗かオプションの設定に失敗                                |
| <code>E_NOTSOCK</code>       | 1007 | ディスクリプタが正しくありません                                              |
| <code>E_INVALID</code>       | 1008 | パラメータ <code>flags</code> が不正です                                |
| <code>E_OPNOTSUPP</code>     | 1012 | ソケットタイプが <code>SOCK_DGRAM</code> (UDP) ではありません                |
| <code>E_SHUTDOWN</code>      | 1013 | ソケットは shutdown されています                                         |
| <code>E_FAULT</code>         | 1016 | データバッファが不正、 <code>len</code> が小さすぎる、 <code>from</code> が NULL |
| <code>E_NETDOWN</code>       | 1018 | 内部エラーまたは受信がキャンセルされました                                         |
| <code>E_TIMEOUT</code>       | 1023 | 送信がタイムアウトしました                                                 |

---

## close

---

【機能】 ソケットを閉じる

【形式】 `int close(int sockfd)`  
      `sockfd` ソケットディスクリプタ

【戻値】   0       正常終了  
      -1       エラー

【解説】 ソケットを閉じます。close はソケットディスクリプタを切り離します。  
エラーの場合 `ns1_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード        | 値    | 内容                       |
|---------------|------|--------------------------|
| E_SOCKNOTINIT | 1001 | sock_ini() で初期化が行われていません |
| E_NOTSOCK     | 1007 | ディスクリプタが正しくありません         |
| E_NOTCONN     | 1014 | ソケットが未接続です               |
| E_MEMORY      | 1017 | メモリ解放エラー                 |
| E_NETDOWN     | 1018 | 内部エラー                    |

---

## shutdown

---

【機能】 send、recv 呼び出しを無効にする

【形式】 int shutdown(int sockfd, int howto)

sockfd ソケットディスクリプタ

howto どの型の操作を禁止するかを示すフラグ。現バージョンでは SHUT\_RDWR 固定です。

【戻値】 0 正常終了

-1 エラー

【解説】 ソケットでの送信、受信のいずれか、または両方を禁止します。このバージョンでは送受信両方の禁止のみサポートしています。

エラーの場合 ns\_l\_get\_err() 関数を使用してエラーの詳細を調べることができます。

| エラーコード        | 値    | 内容                       |
|---------------|------|--------------------------|
| E_SOCKNOTINIT | 1001 | sock_ini() で初期化が行われていません |
| E_NOTSOCK     | 1007 | ディスクリプタが正しくありません         |
| E_INVALID     | 1008 | 1 つまたはそれ以上のパラメータが不正です    |
| E_NOTCONN     | 1014 | ソケットが未接続です               |
| E_NETDOWN     | 1018 | 内部エラー                    |

---

## setsockopt

---

【機能】 ソケットオプションを設定する

【形式】 `int setsockopt(int sockfd, int level, int optname, const void *optval, socklen_t optlen)`

`sockfd` ソケットディスクリプタ

`level` プロトコル依存コード (SO\_SOCKET 固定)

`optname` 値を設定するソケットオプション

`optval` オプションに指定する値を持つバッファへのポインタ

`optlen` `optval` のサイズ

【戻値】 0 正常終了

-1 エラー

【解説】 以下のソケットオプションを設定できます。

| level     | optname      | 説明               | データ型 |
|-----------|--------------|------------------|------|
| SO_SOCKET | SO_BROADCAST | ブロードキャストの送信を許可する | int  |
|           | SO_RCVBUF    | 受信バッファサイズを設定する   | int  |
|           | SO_SNDBUF    | 送信バッファサイズを設定する   | int  |
|           | SO_RCVTIMEO  | 受信タイムアウトを設定する    | TMO  |
|           | SO_SNDTIMEO  | 送信タイムアウトを設定する    | TMO  |
|           | SO_CLSTIMEO  | クローズタイムアウトを設定する  | TMO  |
|           | IP_TOS       | ToS フィールドの値を設定する | int  |
|           | IP_TTL       | TTL フィールドの値を設定する | int  |

SO\_RCVBUF、SO\_SNDBUF のオプションを使用するときには `connect` および `listen` を呼び出す前に設定する必要があります。

SO\_SNDTIMEO、SO\_RCVTIMEO、SO\_CLSTIMEO オプションで指定する値は、BSD ソケットインタフェースでは `struct timeval` 型ですが、本 API では TMO 型に簡略化してあります。TMO 型でも、これに TMO\_POL(0) によってポーリングは指定できません。0 を指定すると、TMO\_FEVR(-1) と同じタイムアウトなしに設定されます。

エラーの場合、次ページの `getsockopt` と同様に、`nsl_get_err()` 関数を使用してエラーの詳細を調べることができます。

---

## getsockopt

---

【機能】 ソケットオプションを取得する

【形式】 `int getsockopt(int sockfd, int level, int optname, void *optval, socklen_t *optlen)`

`sockfd` ソケットディスクリプタ

`level` プロトコル依存コード (SO\_SOCKET 固定)

`optname` 値を取得するソケットオプション

`optval` オプションの値を受け取るバッファへのポインタ

`optlen` `optval` のサイズ

【戻値】 0 正常終了

-1 エラー

【解説】 以下のソケットオプションに現在設定されている値を取得できます。

| level     | optname      | 説明               | データ型 |
|-----------|--------------|------------------|------|
| SO_SOCKET | SO_BROADCAST | ブロードキャストの送信許可    | int  |
|           | SO_RCVBUF    | 受信バッファサイズを取得する   | int  |
|           | SO_SNDBUF    | 送信バッファサイズを取得する   | int  |
|           | SO_RCVTIMEO  | 受信タイムアウトを取得する    | TMO  |
|           | SO_SNDTIMEO  | 送信タイムアウトを取得する    | TMO  |
|           | SO_CLSTIMEO  | クローズタイムアウトを取得する  | TMO  |
|           | SO_TYPE      | ソケットタイプ          | int  |
|           | IP_TOS       | ToS フィールドの値を取得する | int  |
|           | IP_TTL       | TTL フィールドの値を取得する | int  |

エラーの場合 `ns_l_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード          | 値    | 内容                                                                        |
|-----------------|------|---------------------------------------------------------------------------|
| E_SOCKETNOTINIT | 1001 | <code>sock_ini()</code> で初期化が行われていません                                     |
| E_NOTSOCK       | 1007 | ディスクリプタが正しくありません                                                          |
| E_INVALID       | 1008 | <code>level</code> の値が不正です                                                |
| E_NOPROTOOPT    | 1015 | 指定したオプションは未サポートです                                                         |
| E_FAULT         | 1016 | <code>optval</code> ポインタが NULL になっているか <code>optlen</code> パラメータの値が小さすぎます |

---

## nsi\_get\_err

---

【機能】 ソケット関数のエラーの詳細を取り出す

【形式】 ER nsi\_get\_err(void)

【戻値】 エラーコード

【解説】 ソケット関数でエラーになった場合のエラーコードが返ります。  
エラーが発生していない場合はE\_OK が返ります。

---

## inet\_addr

---

【機能】 IPv4 のアドレス文字列を 32bit バイナリアドレスへ変換する

【形式】 `in_addr_t inet_addr(const char *strptr)`  
strptr “.” で区切られた IP アドレス文字列

【戻値】 正常終了した場合、32bit のインターネットアドレス値が返ります。  
strptr に正しい値が入っていない場合、`INADDR_NONE` が返ります。

【解説】 “192.168.0.1” のような文字列をネットワークバイトオーダーの 32bit の IPv4 アドレスに変換します。

---

## inet\_ntoa

---

【機能】 32bit バイナリアドレスをドット付き 10 進文字列に変換する

【形式】 `char *inet_ntoa(struct in_addr inaddr)`  
inaddr ネットワークアドレスを表す in\_addr 構造体

【戻値】 正常終了した場合、ドット付き 10 進文字列へのポインタが返ります。  
エラーの場合、NULL が返ります。

【解説】 ネットワークバイトオーダーの 32bit の IPv4 アドレスを” 192.168.0.1” のような 10 進文字列に変換します。

この関数で得られる文字列のポインタはソケット内部で確保され、リエントラントではありませんので、他のソケットから呼び出しが行われる前にデータをコピーしてください。

---

## inet\_aton

---

【機能】 IPv4 のアドレス文字列を 32bit バイナリアドレスへ変換する

【形式】 `int inet_aton(const char *src, struct in_addr *dst)`  
src      ”.” で区切られた IP アドレス文字列  
dst      ネットワークアドレスを表す in\_addr 構造体へのポインタ

【戻値】 正常終了した場合、1 が返ります。  
src に正しい値が入っていない場合、0 が返ります。

【解説】 “192.168.0.1” のような文字列をネットワークバイトオーダーの 32bit の IPv4 アドレスに変換します。

---

## inet\_pton

---

【機能】 IPv4 および IPv6 のアドレス文字列をバイナリアドレスへ変換する

【形式】 `int inet_pton(int af, const char *src, void *dst)`

af        アドレスファミリ

src       IPv4 または IPv6 アドレス文字列

dst       ネットワークアドレス構造体へのポインタ

【戻値】 正常終了した場合、1 が返ります。

src に正しい値が入っていない場合、0 が返ります。

入力したアドレスファミリが未対応の場合、-1 が返ります。

【解説】 af は IPv4 を使用する場合は AF\_INET を、IPv6 を使用する場合は AF\_INET6 を指定します。アドレスファミリが AF\_INET の場合、“192.168.0.1” のような IPv4 ネットワークアドレス文字列を struct in\_addr に変換します。アドレスファミリが AF\_INET6 の場合、“fe80::1034:56ff:fe78:9a9b” のような IPv6 ネットワークアドレス文字列を struct in6\_addr に変換します。

エラーの場合 `ns_l_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード        | 値    | 内容                        |
|---------------|------|---------------------------|
| E_AFNOSUPPORT | 1002 | 指定されたアドレスファミリがサポートされていません |

---

## inet\_ntop

---

【機能】 ネットワークアドレス構造体で表しているバイナリアドレスを文字列へ変換する

【形式】 `const char *inet_ntop(int af, const void *src, char *dst, socklen_t size)`

af        アドレスファミリ

src       ネットワークアドレス構造体へのポインタ

dst       IPv4 または IPv6 アドレス文字列を格納するバッファへのポインタ

size      文字列を格納するバッファのサイズ

【戻値】 正常終了した場合、ネットワークアドレス文字列を格納するバッファへのポインタ dst が返ります。

エラーの場合、NULL が返ります。

【解説】 af は IPv4 を使用する場合は AF\_INET を、IPv6 を使用する場合は AF\_INET6 を指定します。アドレスファミリが AF\_INET の場合、ネットワークバイトオーダーの IPv4 アドレス (struct in\_addr) を文字列に変換します。アドレスファミリが AF\_INET6 の場合、ネットワークバイトオーダーの IPv6 アドレス (struct in6\_addr) を文字列に変換します。

エラーの場合 `nsi_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード        | 値    | 内容                        |
|---------------|------|---------------------------|
| E_AFNOSUPPORT | 1002 | 指定されたアドレスファミリがサポートされていません |
| E_NOSPC       | 1024 | アドレス文字列がバッファサイズを超えています    |

---

## gethostbyname

---

【機能】 ホスト名からホスト情報を検索する

【形式】 `struct hostent *gethostbyname(const char *hostname)`  
           hostname           ホスト名が入った文字列へのポインタ

【戻値】 正常に終了した場合、`struct hostent` 型のポインタが返ります。  
           エラーの場合、`NULL` が返ります。

【解説】 `hostent` 構造体は以下のようになっています。

```
struct hostent {
    char *h_name;      ホスト名
    char **h_aliases;  エイリアス名へのポインタ配列へのポインタ
    int   h_addrtype;  ホストアドレスのタイプ (AF_INET)
    int   h_length;    アドレスの大きさ (4)
    char **h_addr_list; アドレスへのポインタ配列へのポインタ
};
```

この関数で返される `hostent` 構造体は固定的に確保され、リエントラントではありませんので、複数のタスクから同時に操作することはできません。API の実行終了を待ってから操作する必要があります。

エラーの場合 `ns_l_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード        | 値    | 内容                                    |
|---------------|------|---------------------------------------|
| E_SOCKNOTINIT | 1001 | <code>sock_ini()</code> で初期化が行われていません |
| E_FAULT       | 1016 | 文字列が不正です                              |
| E_NETDOWN     | 1018 | ホストが見つからないまたは DNS との通信が失敗             |

---

## getpeername

---

【機能】 ソケットのリモートプロトコルアドレスを返す

【形式】 `int getpeername( int sockfd, struct sockaddr *peeraddr, socklen_t *addrlen)`

sockfd                      ソケットディスクリプタ

peeraddr                    プロトコルアドレスを格納する構造体のポインタ

addrlen                    構造体のサイズを格納するポインタ

【戻値】    0            正常終了

          -1            エラー

【解説】 接続中のリポートホストのアドレス情報を取得できます。getpeername 関数は接続されているソケットでのみ使用可能です。

エラーの場合 `ns_l_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード        | 値    | 内容                                     |
|---------------|------|----------------------------------------|
| E_SOCKNOTINIT | 1001 | sock_ini() で初期化が行われていません               |
| E_NOTSOCK     | 1007 | ディスクリプタが正しくありません                       |
| E_INVALID     | 1008 | addrlen のサイズが小さすぎるか、peeraddr が NULL です |
| E_NOTCONN     | 1014 | ソケットが接続されていません                         |

---

## getsockname

---

【機能】 ソケットのローカルプロトコルアドレスを返す

【形式】 `int getsockname (int sockfd, struct sockaddr *localaddr, socklen_t *addrlen)`

sockfd                  ソケットディスクリプタ

localaddr              プロトコルアドレスを格納する構造体のポインタ

addrlen                構造体のサイズを格納するポインタ

【戻値】    0            正常終了

          -1            エラー

【解説】 `getsockname` はバインドされているか接続されているソケットで使用されます。  
エラーの場合 `ns_l_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード        | 値    | 内容                                                                 |
|---------------|------|--------------------------------------------------------------------|
| E_SOCKNOTINIT | 1001 | <code>sock_ini()</code> で初期化が行われていません                              |
| E_NOTSOCK     | 1007 | ディスクリプタが正しくありません                                                   |
| E_INVALID     | 1008 | <code>addrlen</code> のサイズが小さすぎるか、 <code>localaddr</code> が NULL です |

---

## getaddrinfo

---

【機能】 ホスト名とポート番号に該当するネットワークアドレス構造体を返す

【形式】 `int getaddrinfo(const char *nodename, const char *servname,  
                          const struct addrinfo *hints, struct addrinfo **res)`

|                       |                                 |
|-----------------------|---------------------------------|
| <code>nodename</code> | ホスト名、または数値のホストアドレスが入った文字列へのポインタ |
| <code>servname</code> | ポート番号が入った文字列へのポインタ              |
| <code>hints</code>    | ソケット情報を提供する構造体へのポインタ            |
| <code>res</code>      | アドレス情報を格納する構造体のリンクリストへのポインタ     |

【戻値】

|              |                                    |
|--------------|------------------------------------|
| 0            | 正常終了                               |
| EAI_NONAME   | nodename および servname が NULL       |
| EAI_FAMILY   | アドレスファミリが未対応                       |
| EAI_SOCKTYPE | ソケットタイプが未対応                        |
| EAI_SERVICE  | nodename または servname に正しい値が入っていない |
| EAI_NODATA   | ドメイン名から IP アドレスの取得に失敗              |
| EAI_FAIL     | プロトコルスタック未初期化                      |

【解説】 `addrinfo` 構造体は以下のようになっています。

```
struct addrinfo {
    int      ai_flags;          AI_PASSIVE, AI_NUMERICHOST フラグ
    int      ai_family;        アドレスファミリ
    int      ai_socktype;      ソケットタイプ
    int      ai_protocol;      プロトコルタイプ
    socklen_t ai_addrlen;      ai_addr の長さ
    char     *ai_canonname;     ノード名の正規名
    struct sockaddr *ai_addr;   バイナリアドレス
    struct addrinfo *ai_next;   次の構造体へのポインタ
};
```

`hints` パラメータにより次のようにソケット情報を渡します。

`ai_flags` に `AI_PASSIVE` フラグがセットされ、`nodename` が NULL である場合は、アドレス構造体のネットワークアドレスが 0 (Any address) に設定されます。`nodename` が "192.168.1.23" のような数値のホストアドレスが入った文字列の場合は、

ai\_flags に AI\_NUMERICHOST フラグをセットする必要があります。

ai\_family には、IPv4 を使用する場合は AF\_INET を、IPv6 を使用する場合は AF\_INET6 を指定します。AF\_UNSPEC を指定した場合、または、hints が NULL の場合、AF\_INET が選択されます。

ai\_socktype は、SOCK\_DGRAM または SOCK\_STREAM を設定します。ai\_socktype が 0、または、hints が NULL の場合、SOCK\_STREAM が選択されます。

ai\_protocol と ai\_addrlen には 0 を設定してください。ai\_canonname、ai\_addr、および ai\_next には NULL を設定してください。

servname には、” 123” のような数値のポート番号が入った文字列を渡してください。NULL の場合、ポート番号が 0 に設定されます。

この関数で返されるアドレス構造体は固定的に確保され、リエントラントではありませんので、複数のタスクから同時に操作することはできません。API の実行終了を待ってから操作する必要があります。

---

## freeaddrinfo

---

【機能】 getaddrinfo 関数による割り当てられたリソースを解放する

【形式】 void freeaddrinfo(struct addrinfo \*ai)  
ai        addrinfo 構造体へのポインタ

【戻値】 なし

【解説】 現在、getaddrinfo は固定メモリ領域を使用しているので、本関数は何もしません。

---

## select

---

【機能】 入出力の多重化

【形式】 `int select(int maxfdp1, fd_set *rset, fd_set *wset, fd_set *eset, const TMO *timeout)`

`maxfdp1` 検査するソケットディスクリプタ番号の最大値+1

`rset` 読み出し可能を検出するソケットディスクリプタ集合へのポインタ

`wset` 書き込み可能を検出するソケットディスクリプタ集合へのポインタ

`eset` 例外状態を検出するソケットディスクリプタ集合へのポインタ

`timeout` タイムアウト値

【戻値】 正常に終了した場合、準備ができているソケットディスクリプタの個数が返ります。タイムアウトの場合は 0 が、エラーの場合は -1 が返ります。

【解説】 `select` を実行したタスクは待ち状態となり、以下のいずれかのイベントが発生した場合に待ちが解除されます。

1. 読み出しの場合

a) データを受信した

b) コネクションの読み出し側がクローズした

2. 書き込みの場合

a) 書き込みを行うバッファの準備ができた

例外状態の検出は未サポートです。 `eset` には NULL ポインタを設定してください。  
`timeout` に NULL を指定した場合、タイムアウトなし (TMO\_FEVR) と同じと見なされます。  
ポーリング (TMO\_POL) の指定は可能です。

エラーの場合 `ns1_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード        | 値    | 内容                                                                      |
|---------------|------|-------------------------------------------------------------------------|
| E_SOCKNOTINIT | 1001 | <code>sock_ini()</code> で初期化が行われていません                                   |
| E_NOTSOCK     | 1007 | ディスクリプタが正しくありません                                                        |
| EINVAL        | 1008 | パラメータが不正です                                                              |
| E_FAULT       | 1016 | <code>select</code> を呼ぶタスクの ID が不正であるか、プロトコルスタック内部リソースをアクセスするための排他処理が失敗 |
| E_NOSPC       | 1024 | <code>select</code> 管理テーブルのエントリ数が不足                                     |

---

## ioctl

---

【機能】 ソケットを制御する

【形式】 `int ioctl(int sockfd, unsigned long cmd, void *data)`

`sockfd` ソケットディスクリプタ

`cmd` ソケットで実行されるコマンドのコード

`data` コマンドに対応するデータを格納するポインタ

【戻値】 0 正常終了

-1 エラー

【解説】 コマンドによってソケットに関連するパラメータを取得、または設定します。次のコマンドをサポートします。

| コマンド     | 説明                                                                                                                                                                                                                           | データ型 |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| FIONREAD | <code>recv()</code> または <code>recvfrom()</code> を呼ぶと読み出せる最大のデータサイズを返します。ソケットタイプが <code>SOCK_DGRAM</code> の場合には、キューイングされている最初の UDP データグラムのデータサイズを返します。ソケットタイプが <code>SOCK_STREAM</code> の場合には、TCP 受信バッファに格納されているデータのサイズを返します。 | int  |

エラーの場合 `ns_l_get_err()` 関数を使用してエラーの詳細を調べることができます。

| エラーコード        | 値    | 内容                                            |
|---------------|------|-----------------------------------------------|
| E_SOCKNOTINIT | 1001 | <code>sock_ini()</code> で初期化が行われていません         |
| E_NOTSOCK     | 1007 | ディスクリプタが正しくありません                              |
| E_INVALID     | 1008 | <code>data</code> が NULL か、コマンドが未対応か、ソケットが未生成 |
| E_OPNOTSUPP   | 1012 | 未対応のソケットタイプ                                   |
| E_NOTCONN     | 1014 | ソケットが未接続です                                    |
| E_FAULT       | 1016 | 内部エラー                                         |
| E_NETDOWN     | 1018 | プロトコルスタック未初期化                                 |
| E_CONNRESET   | 1021 | リセットによりソケットは切断されました                           |

## NORTi Socket Interface ユーザーズガイド

---

|           |                                                               |
|-----------|---------------------------------------------------------------|
| 株式会社ミスポ   | <a href="http://www.mispo.co.jp/">http://www.mispo.co.jp/</a> |
| 一般のお問い合わせ | <a href="mailto:sales@mispo.co.jp">sales@mispo.co.jp</a>      |
| 技術サポートご依頼 | <a href="mailto:norti@mispo.co.jp">norti@mispo.co.jp</a>      |

---