RFC1813 準拠 NFS クライアント

NFS for NORTi

ユーザーズガイド

2006 年 1 月 16 日 第 2 版

株式会社ミスポ

はじめに

NFS for NORTi は、リアルタイム OS「NORTi」上で動作する、Network File System Protocol Version 3 (RFC1813)に準拠した NFS クライアントです。ここでは、NFS for NORTi の使用方法と、NFS サーバーの設定方法について説明します。ご使用になる前に必ずご一読ください。

目次

第 1 章 導入

1.1 フォルダ構成

NFS for NORTi のフォルダ構成は、次の通りです。

```
NORTi
├ NOFNFS
│  ├ DOC                  ドキュメント
│  │  ├ nofnfs.pdf
│  │  └ NFS_XXX_YYY_ZZZ.txt
│  ├ INC                  ヘッダファイル
│  │  ├ nfscfg.h
│  │  ├ nfssys.h
│  │  └ nofnfs.h
│  ├ LIB                  ライブラリ
│  │  └ XXX
│  │     └ YYY
│  └ SRC                  ソースファイル
│     ├ nfstime.c
│     ├ nofnfs.c
│     └ nofvfs.c
└ SMP                     評価ボード用サンプルプログラム
   └ XXX
      └ BBB
```

上記の *XXX* は対応プロセッサ名略称(例：SH, ARM, etc.)、*YYY* は対応コンパイラ名略称(例：SHC7, SHC8, SHC9, etc.)、*BBB* は評価ボード名(例：MS7750S, SE27CPF etc.)です。実際のフォルダ名は、DOC フォルダの補足説明書を参照してください。

1.2 ファイル構成

ドキュメント

nofnfs.pdf

NFS for NORTi ユーザーズガイド(本書)です。ファイル構成、API 仕様、制限事項など、各処理系(プロセッサやコンパイラ)に依存しない共通の事柄を説明しています。

NFS_*XXX*_*YYY*_*ZZZ*.txt

NFS for NORTi 補足説明書です。処理系に依存する部分の説明や修正履歴を記述しています。ファイル名の *XXX*、*YYY* の部分は処理系によって異なります。*ZZZ* はバージョンです。

ヘッダファイル

nofnfs.h

NFS for NORTi ヘッダファイルです。NFS for NORTi を使用する全てのソースファイルで #include してください。NFS for NORTi を使用する為に必要な関数宣言、構造体、マクロ定義が記述されています。

nfscfg.h

NFS for NORTi コンフィグレーションヘッダファイルです。アプリケーションの１つのソースファイルでのみ#include してください。#include 文の上に本ヘッダ内に定義されるマクロの値を定義することにより、コンフィグレーションを行うことができます。

nfssys.h

NFS for NORTi 内部定義ヘッダです。NFS for NORTi 内部で使用されます。アプリケーションで直接#include する必要はありません。

ライブラリ

NFS for NORTi ライブラリは、プロセッサ、コンパイラ、エンディアン、デバッグ情報の有無によって異なるライブラリがそれぞれ用意されています。ご使用の開発環境に必要なライブラリをアプリケーションにリンクしてください。

ファイル名の命名規則は次の通りです。

nfd*XXXX*.lib　　デバッグ情報付きライブラリ
nfs*XXXX*.lib　　デバッグ情報なしライブラリ

*XXXX*の部分はプロセッサやエンディアンによって異なります。例えば、ルネサス SuperH 用のライブラリファイル名は次のようになります。

デバッグ情報付きライブラリ

nfdsh1.lib	SH-1 ビッグエンディアン
nfdsh2.lib	SH-2 ビッグエンディアン
nfdsh3b.lib	SH-3 ビッグエンディアン
nfdsh3l.lib	SH-3 リトルエンディアン
nfdsh4b.lib	SH-4 ビッグエンディアン
nfdsh4l.lib	SH-4 リトルエンディアン

デバッグ情報なしライブラリ

nfssh1.lib	SH-1 ビッグエンディアン
nfssh2.lib	SH-2 ビッグエンディアン
nfssh3b.lib	SH-3 ビッグエンディアン
nfssh3l.lib	SH-3 リトルエンディアン
nfssh4b.lib	SH-4 ビッグエンディアン
nfssh4l.lib	SH-4 リトルエンディアン

ソースファイル

NFS for NORTi はライブラリ化されていますので、通常は以下のソースファイルをコンパイルしてアプリケーションにリンクする必要はありません。

nofnfs.c
NFS for NORTi ソースファイル

nofvfs.c
NFS for NORTi と NORTi File System Version 4 を併用する場合にコールされるリダイレクト関数

nfstime.c
ANSI C 準拠の時間関数

1.3 主な仕様

プロトコル

NFS for NORTi が対応するプロトコルバージョンは次の通りです。

- MOUNT Protocol Version 3
- Network File System Protocol Version 3
- PCNFS Protocol Version 2
- Portmap Protocol Version 2
- Remote Procedure Call Protocol Version 2

※ PCNFS プロトコルは AUTH プロシジャのみ対応

※ PORTMAP プロトコルは GETPORT プロシジャのみ対応

対応 NFS サーバー

NFS クライアントである NFS for NORTi が対応する NFS サーバーは次の通りです。

- Microsoft Windows Services for UNIX 3.5 日本語版

第 2 章 関数

2.1 NFS 関数

nfs_ini　　NFS クライアントの初期化

形式　ER nfs_ini(void)

戻値

	E_OK	正常終了
	その他	TCP/UDP 通信端点、タスク、メールボックス、メモリプールを生成するシステムコールまたは sta_tsk のエラーコード

解説　NFS クライアントの初期化を行います。NFS for NORTi が使用するタスク、メールボックス、メモリプール、TCP/UDP の通信端点を生成し、NFS クライアントタスクを起動します。nfs_ini は、TCP/IP プロトコルスタック初期化関数 tcp_ini を実行した後に 1 回実行してください。nfs_ini を実行すると、他の全ての NFS for NORTi 関数が使用可能となります。

例　TCP/IP プロトコルスタック初期化後に NFS クライアントを初期化します。

```
TASK MainTask (void)
{
    ER ercd;

    /* TCP/IP プロトコルスタック初期化 */
    ercd = tcp_ini();
    if (ercd < 0)
        goto END;

    /* NFS for NORTi 初期化 */
    ercd = nfs_ini();
    if (ercd != E_OK)
        goto END;
            :
```

mount　　マウント

形式　ER mount(const char *user, const char *pass, const char *server, const char *share, const char *drive, UW param)

引数

user	ユーザー名
pass	パスワード
server	NFS サーバーIP アドレス
share	共有ディレクトリ名
drive	ドライブ名
param	NFS サーバー依存パラメータ

戻値

E_OK	正常終了
EV_MOUNT	二重マウント
EV_FPAR	無効な IP アドレスまたは共有ディレクトリ名を指定
EV_RPC	リモートプロシジャコール要求が拒否または無視された
EV_PMAP	ポートマッパーによるポート番号取得に失敗
EV_PCNFS	PCNFS 認証に失敗
EV_MOUNTP	サーバーがマウントを拒否
EV_NFS	サーバーのファイルシステム情報取得に失敗
EV_DRVNAME	ドライブ名が不正

説明　NFS サーバーによって共有されたディレクトリをマウントします。共有ディレクトリは、drive で指定したドライブ名に関連付けられます。mount 実行後 unmount を実行するまで、割り当てたドライブに対してファイル/ディレクトリ操作関数を実行することができます。

user と pass には NFS サーバーで設定したユーザー名とパスワードを指定してください。

server には"192.168.0.11"のように NFS サーバーの IP アドレス文字列を指定してください。

share には"/SHARE"のように'/'(スラッシュ)+共有ディレクトリ名を指定してください。

drive には"A:"、"B:"のように共有ディレクトリに割り当てるドライブ名を、(アルファベット a～z または A～Z)＋':'(コロン)で指定してください。大文字小文字

の区別はありません。
param は現在未使用です。0 を設定してください。

同時に複数の共有ディレクトリをマウントすることはできません。マウントした共有ディレクトリとは別の共有ディレクトリをマウントする場合、現在のマウントをアンマウントしてからマウントしてください。
NFS サーバー側のユーザー名とパスワードの設定方法については、『Microsoft Windows Services for UNIX 3.5』の章を参照してください。

例　ユーザー名 pcnfslo、パスワード pcnfspw で、IP アドレスが 192.168.0.11 である NFS サーバー上の共有ディレクトリ TEMP をドライブ C:に割り当てます。

```
TASK MainTask(void)
{
    ER ercd;

    /* IP & Ethernet Address の読み出し */
    read_ip_addr();
    read_ethernet_addr();

    /* TCP/IP プロトコルスタック初期化 */
    ercd = tcp_ini();
    if (ercd < 0)
        goto END;

    /* NFS for NORTi 初期化 */
    ercd = nfs_ini();
    if (ercd != E_OK)
        goto END;

    /* マウント */
    ercd = mount("pcnfslo", "pcnfspw", "192.168.0.11", "/TEMP", "C:", 0);
    if (ercd != E_OK)
        goto END;
            :
```

unmount　　アンマウント

形式　ER unmount(const char *drive)

引数　drive　　ドライブ名

戻値　E_OK　　正常終了
　　EV_UNMOUNT　　マウントしていない
　　EV_OPENED　　開いているファイルまたはディレクトリが存在する
　　EV_RPC　　リモートプロシジャコール要求が拒否または無視された
　　EV_DRVNAME　　ドライブ名が不正

説明　mount でマウントしたドライブをアンマウントします。これにより、共有ディレクトリに割り当てた drive が未使用となります。NFS サーバーとの通信プロトコルに TCP(既定値)を選択した場合、TCP 接続を切断します。

例　ドライブ C:をアンマウントします。

```
TASK MainTask(void)
{
    ER ercd;
        :
    /* マウント */
    ercd = mount("pcnfslo", "pcnfspw", "192.168.0.11", "/TEMP", "C:", 0);
    if (ercd != E_OK)
        goto END;
            :
    (ドライブ C:に対するファイル、ディレクトリ操作)
            :
    /* アンマウント */
    ercd = unmount("C:");
    if (ercd != E_OK)
        goto END;
            :
```

2.2 ファイル操作関数

fopen　　ファイルのオープン

形式　FILE *fopen(const char *path, const char *mode)

引数　path　　ファイル名
　　mode　　オープンモード

戻値　NULL 以外　　ファイルポインタ
　　NULL　　エラー

説明　path で指定したファイルを開きます。
path には、C:¥aaaa¥bbbb¥nnnnnnnn.eee 形式のドライブ名やルートからのディレクトリ名を含むフルパスでファイル名を指定してください。path の(ドライブ名):¥以降に付加するパスの最大長は、ヘッダ nofnfs.h に定義される NFS_MAX_PATH(1024)です。path に含まれるディレクトリ名やファイル名の最大長は同 NFS_MAX_NAME (255)です。但し、実際に使用できるパスやファイル名の最大長は、NFS サーバーのファイルシステムに依存します。カレントディレクトリの管理は行っていません。従って、..¥..¥のような相対パスによる指定はできません。(ドライブ名):¥は省略することができます。
mode に指定するオープンモードは以下の３つのいずれかです。

″r″　　読み込みモード
″w″　　書き込みモード
″a″　　追加書き込みモード

同一のファイルを同時に 2 回以上 fopen できるのは″r″モードの場合のみです。

また、上記モードに以下を付加できます。

″+″　　更新モード
″b″　　バイナリモード

"t"で指定するテキストモードは未サポートです。従って、"b"を指定しない場合もバイナリモードとなります。

同時にオープンすることができるファイル(とディレクトリ)数の合計は、コンフィグレーションマクロ NFS_NFILE の値で決定されます。既定値は 8 です。

例　　共有ディレクトリにある test.txt を読み込みモードで開きます。

```
TASK MainTask(void)
{
    FILE *fp;
        :
    fp = fopen("C:¥¥test.txt", "r");
    if (!fp)
        goto ERR;
        :
```

fclose　　ファイルのクローズ

形式　int fclose(FILE *fp)

引数　fp　　ファイルポインタ

戻値　0　　正常終了
　　EOF　　エラー

説明　fp で指定したファイルを閉じます。ファイルが書き込みモードで開かれており、READ/WRITE バッファに書き込みデータが残っている場合は、NFS サーバー上のファイルにデータを書き込んでからファイルを閉じます。

例　fopen で開いたファイル C:¥test.txt を閉じます。

```
TASK MainTask(void)
{
    FILE *fp;

    fp = fopen("C:¥¥test.txt", "r");
    if (!fp)
        goto ERR;
        :
    fclose(fp);
```

fgetc　　ファイルから 1 文字読み出し

形式　int fgetc(FILE *fp)

引数　fp　　ファイルポインタ

戻値　文字　　読み出した文字
　　EOF　　エラーまたはファイルの終端

説明　fp で指定したファイルから 1 文字読み出します。

例　ファイル test.txt から 10 文字読み出します。

```
TASK MainTask(void)
{
    FILE *fp;
    int i, c;

    fp = fopen("C:¥¥test.txt", "r");
    if (!fp)
        goto ERR;
    for (i = 0; i < 10; i++) {
        c = fgetc(fp);
        if (c == EOF)
            break;
    }
    fclose(fp);
```

fputc　　ファイルへ 1 文字書き込み

形式　int fputc(int c, FILE *fp)

引数

c	書き込む文字
fp	ファイルポインタ

戻値

文字	書き込んだ文字
EOF	エラー

説明　fp で指定したファイルに文字 c を書き込みます。

例　ファイル test.txt に文字'a'を 10 回書き込みます。

```
TASK MainTask(void)
{
    FILE *fp;
    int i, c;

    fp = fopen("C:¥¥test.txt", "w");
    if (!fp)
        goto ERR;
    for (i = 0; i < 10; i++) {
        c = fputc('a', fp);
        if (c != 'a')
            break;
    }
    fclose(fp);
```

fgets　　ファイルから文字列読み出し

形式　char *fgets(char *s, int n, FILE *fp)

引数

	s	データの格納場所
	n	データ格納場所のサイズ
	fp	ファイルポインタ

戻値

	s	正常終了
	NULL	エラーまたはファイルの終端

説明　fp で指定したファイルから 1 行分の文字列を読み出してバッファ s に格納します。読み出しは、fp の現在の読み書き位置から最初の改行文字('¥n')が現れるか、ファイルの終端に達するか、読み出した文字列が n-1 文字になるまで行われ、文字列の最後に NULL 文字('¥0')が付加されます。

例　ファイル test.txt からバッファに最大 29 文字の文字列を連続して読み出します。

```
TASK MainTask(void)
{
    FILE *fp;
    char buf[30], *p;

    fp = fopen("C:¥¥test.txt", "r");
    if (!fp)
        goto ERR;
    for (;;) {
        p = fgets(buf, 30, fp);
        if (!p)
            break;
    }
    fclose(fp);
```

fputs　　ファイルへ文字列書き込み

形式　int fputs(const char *s, FILE *fp)

引数

s	出力する文字列
fp	ファイルポインタ

戻値

0	正常終了
EOF	エラー

説明　fp で指定したファイルへ文字列 s を書き込みます。

例　ファイル test.txt に文字列"abcdefg"を 10 回書き込みます。

```
TASK MainTask(void)
{
    FILE *fp;
    int i, r;

    fp = fopen("C:¥¥test.txt", "w");
    if (!fp)
        goto ERR;
    for (i = 0; i < 10; i++) {
        r = fputs("abcdefg", fp);
        if (r == EOF)
            break;
    }
    fclose(fp);
```

fread　　ファイルからブロック読み出し

形式　size_t fread(void *data, size_t size, size_t n, FILE *fp)

引数

	data	データの格納場所
	size	バイト単位の項目サイズ
	n	読み出す最大項目数
	fp	ファイルポインタ

戻値

	n	読み出した全項目数
	n 未満の数	エラーまたはファイルの終端

説明　fp で指定したファイルから最大 size×n バイトのデータを読み出してバッファ data に格納します。

例　ファイル test.txt から 1024 バイトのデータを 10 回読み出します。

```
static char buf[1024];
TASK MainTask(void)
{
    FILE *fp;
    int i;
    size_t n;

    fp = fopen("C:¥¥test.txt", "r");
    if (!fp)
        goto ERR;
    for (i = 0; i < 10; i++) {
        n = fread(buf, 1, 1024, fp);
        if (n != 1024)
            break;
    }
    fclose(fp);
```

fwrite　　ファイルへブロック書き込み

形式　size_t fwrite(const void *data, size_t size, size_t n, FILE *fp)

引数

	data	書き込むデータへのポインタ
	size	バイト単位の項目サイズ
	n	書き込まれる最大項目数
	fp	ファイルポインタ

戻値

	n	書き込んだ項目数
	n 未満の数	エラー

説明　fp で指定したファイルへバッファ data に格納されている size×n バイトのデータを書き込みます。

例　ファイル test.txt に 1024 バイトのデータを 10 回書き込みます。

```
static char buf[1024];
TASK MainTask(void)
{
    FILE *fp;
    int i;
    size_t n;

    fp = fopen("C:¥¥test.txt", "w");
    if (!fp)
        goto ERR;
    for (i = 0; i < 10; i++) {
        n = fwrite(buf, 1, 1024, fp);
        if (n != 1024)
            break;
    }
    fclose(fp);
```

ftell　　現在のファイル読み書き位置を取得

形式　long ftell(FILE *fp)

引数　fp　　ファイルポインタ

戻値　0 以上の数　　現在のファイル読み書き位置
　　-1　　エラー

説明　fp で指定したファイルの現在のファイル読み書き位置を返します。

例　ファイル test.txt から 1024 バイトのデータを 10 回読み出します。途中でファイルの終端に達したら、読み書き位置を取得します。

```
static char buf[1024];
TASK MainTask(void)
{
    FILE *fp;
    int i;
    size_t n;
    long pos;

    fp = fopen("C:¥¥test.txt", "r");
    if (!fp)
        goto ERR;
    for (i = 0; i < 10; i++) {
        n = fread(buf, 1, 1024, fp);
        if (n != 1024) {
            pos = ftell(fp);
            break;
        }
    }
    fclose(fp);
```

fseek　　ファイル読み書き位置の移動

形式　int fseek(FILE *fp, long offset, int origin)

引数

	fp	ファイルポインタ
	offset	origin からのバイト数
	origin	初期位置

戻値

	0	正常終了
	-1	エラー

説明　fp で指定したファイルの読み書き位置を、origin で指定した位置から offset で指定したバイト数だけ移動します。origin には nofnfs.h に定義されている以下の３つのうちのいずれかを指定してください。

SEEK_SET	ファイルの先頭
SEEK_CUR	現在位置
SEEK_END	ファイルの終端

例　ファイル test.txt に 1024 バイトのデータを書き込み、読み書き位置を先頭に戻してから 1024 バイトのデータを読み込みます。

```
static char buf[1024];
TASK MainTask(void)
{
    FILE *fp;

    fp = fopen("C:¥¥test.txt", "w+");
    if (!fp)
        goto ERR;
    fwrite(buf, 1, 1024, fp);
    fseek(fp, 0, SEEK_SET);
    fread(buf, 1, 1024, fp);
    fclose(fp);
```

fflush　　ファイルのフラッシュ

形式　int fflush(FILE *fp)

引数　fp　　ファイルポインタ

戻値　0　　正常終了
　　EOF　　エラー

説明　fpで指定したファイルのファイルシステムが管理するREAD/WRITEバッファのデータをフラッシュします。バッファ内に書き込みデータが存在する場合は、NFSサーバー上のファイルへデータの転送が行われます。READ/WRITE バッファのサイズは4Kバイトです。

例　ファイルtest.txtに1024バイトのデータを書き込み、READ/WRITEバッファのデータをフラッシュします。

```
static char buf[1024];
TASK MainTask(void)
{
    FILE *fp;

    fp = fopen("C:¥¥test.txt", "w");
    if (!fp)
        goto ERR;
    fwrite(buf, 1, 1024, fp);
    fflush(fp);
    fclose(fp);
```

feof　　ファイルの終端を検出

形式　int feof(FILE *fp)

引数　fp　　ファイルポインタ

戻値　0　　ファイルの終端ではない
　　正の値　　ファイルの終端に達した
　　-1　　エラー

説明　fp で指定したファイルの終端より後ろに対して読み込み操作を行うと、EOF フラグが設定され、feof は正の値(本ファイルシステムでは 1)を返します。それ以外の場合は 0 を、feof の実行エラーが起こった場合は-1 を返します。EOF フラグは clearerr、fseek を実行するとクリアされます。

例　ファイル test.txt からファイルの終端に達するまでデータを読み込みます。

```
static char buf[1024];
TASK MainTask(void)
{
    FILE *fp;

    fp = fopen("C:¥¥test.txt", "r");
    do {
        fread(buf, 1, 1024, fp);
    } while (feof(fp) == 0);
    fclose(fp);
```

ferror　　エラーコードの取得

形式　int ferror(FILE *fp)

引数　fp　　ファイルポインタ

戻値　0　　エラーが発生していない
　　負の値　　エラーコード

説明　fp で指定したファイルの入出力操作で発生したエラーのエラーコードを返します。エラーコードの値と意味は『エラーコード』の章を参照してください。ferror が返すエラーコードは、clearerr を実行するとクリアされます。

例　　ファイル test.txt のエラーコードを取得します。

```
TASK MainTask(void)
{
    FILE *fp;
    int r;

    fp = fopen("C:¥¥test.txt", "r");
    if (!fp)
        goto END;
    r = fputc('a', fp);
    if (r == EOF)
        r = ferror(fp);    /* r = -101(EV_FMODE) */
            :
```

clearerr　　エラーコードのクリア

形式　void clearerr(FILE *fp);

引数　fp　　ファイルポインタ

戻値　無し

説明　fp で指定したファイルの入出力操作で発生したエラーのエラーコードをクリアします。またファイルに EOF フラグが設定されている場合は EOF フラグをクリアします。

例　ファイル test.txt からファイルの終端に達するまでデータを読み込み、EOF フラグをクリアします。

```
TASK MainTask(void)
{
    FILE *fp;
    int c;

    fp = fopen("C:¥¥test.txt", "r");
    if (!fp)
        goto END;
    do {
        c = fgetc(fp);
    } while (feof(fp) == 0);
    clearerr(fp);       /* EOF フラグをクリア*/
    fclose(fp);
```

remove　　ファイルの削除

形式　int remove(const char *path)

引数　path　　ファイル名

戻値　0　　正常終了
　　-1　　エラー

説明　path で指定したファイルを削除します。ファイル名の指定方法は fopen と同様です。開いているファイルや存在しないファイルを指定するとエラーになります。

例　ファイル test.txt を削除します。

```
TASK MainTask(void)
{
    int r;

    r = remove("C:¥test.txt");
    if (r != 0)
        goto END;
```

rename　　ファイル名の変更

形式　int rename(const char *oldname, const char *newname)

引数

	oldname	古いファイル名
	newname	新しいファイル名

戻値

	0	正常終了
	-1	エラー

説明　oldname で指定したファイルのファイル名を、newname で指定したファイル名に変更します。古いファイル名の指定方法は fopen と同様です。新しいファイル名の(ドライブ名):¥は省略することができます。開いているファイルや存在しないファイルを指定するとエラーになります。

例　ファイル test.txt のファイル名を、sss.txt に変更します。

```
TASK MainTask(void)
{
    int r;

    r = rename("C:¥test.txt", "sss.txt");
    if (r != 0)
        goto END;
```

stat　　ファイルの状態情報の取得

形式　int stat(const char *path, struct stat *buf)

引数

	path	ファイルまたはディレクトリ名
	buf	ファイル情報の格納場所

戻値

	0	正常終了
	-1	エラー

説明　path で指定したファイルまたはディレクトリの状態情報を取得して stat 構造体 buf に格納します。ファイルまたはディレクトリ名の指定方法は fopen と同様です。存在しないファイルまたはディレクトリを指定するとエラーになります。stat 構造体は nofnfs.h に以下のように定義されています。

```
struct stat {
    unsigned long st_mode;      ファイルの種類と保護モードビット
    unsigned long st_ino;       ファイル ID
    unsigned long st_dev;       ファイルシステム ID
    unsigned long st_rdev;      ファイルのデバイス情報
    unsigned long st_nlink;     ハードリンクの数
    unsigned long st_uid;       ファイル所有者のユーザーID
    unsigned long st_gid;       ファイルのグループ ID
    long st_size;               ファイルサイズ(バイト)
    time_t st_atime;            最終アクセス時刻
    time_t st_mtime;            最終更新時刻
    time_t st_ctime;            属性の最終更新時刻
};
```

stat は NFS サーバー上のファイルの状態情報を取得します。従って、書き込みモードで開いたファイルの書き込み後のサイズや各種時刻が stat の結果に反映されるのは、fflush や fclose を実行して READ/WRITE バッファのデータをサーバー上のファイルに転送した後になります。

ファイルの種類は nofnfs.h に定義されている以下のマクロの引数 m に st_mode を指定して判別してください。マクロは該当するファイル種類のとき TRUE を返します。

マクロ名	説明
S_ISSOCK(m)	ソケット
S_ISLNK(m)	シンボリックリンク
S_ISREG(m)	通常のファイル
S_ISBLK(m)	ブロック型特殊ファイル
S_ISDIR(m)	ディレクトリ
S_ISCHR(m)	文字型特殊ファイル
S_ISFIFO(m)	パイプまたは FIFO

ファイルの保護モードは以下のマクロと st_mode の AND(論理積)を求めて判別してください。

マクロ名	値(8 進数)	説明
S_ISUID	0004000	ユーザーID を実行時に設定
S_ISGID	0002000	グループ ID を実行時に設定
S_ISVTX	0001000	ファイルの場合はキャッシュ禁止フラグ ディレクトリの場合は制限付きの削除フラグ
S_IRUSR	0000400	所有者の読み出し許可
S_IWUSR	0000200	所有者の書き込み許可
S_IXUSR	0000100	ファイルの場合は所有者の実行許可 ディレクトリの場合は所有者の検索許可
S_IRGRP	0000040	グループの読み出し許可
S_IWGRP	0000020	グループの書き込み許可
S_IXGRP	0000010	ファイルの場合はグループの実行許可 ディレクトリの場合はグループの検索許可
S_IROTH	0000004	その他の読み出し許可
S_IWOTH	0000002	その他の書き込み許可
S_IXOTH	0000001	ファイルの場合はその他の実行許可 ディレクトリの場合はその他の検索許可

例　　ファイル test.txt の状態情報を取得し、以下の形式の文字列をバッファ combuf に作成します。

```
-rwxrwxrwx   1 100 100          40 Wed Jul 27 18:59:17 2005
```

```
TASK MainTask(void)
{
    int r;
    struct stat st;
    char buf[26], combuf[100];

    r = stat("C:¥test.txt", &st);
    if (r != 0)
        goto END;

    strcpy(combuf, S_ISDIR(st.st_mode) ? "d" : "-");
    strcat(combuf, (st.st_mode & S_IRUSR) ? "r" : "-");
    strcat(combuf, (st.st_mode & S_IWUSR) ? "w" : "-");
    strcat(combuf, (st.st_mode & S_IXUSR) ? "x" : "-");
    strcat(combuf, (st.st_mode & S_IRGRP) ? "r" : "-");
    strcat(combuf, (st.st_mode & S_IWGRP) ? "w" : "-");
    strcat(combuf, (st.st_mode & S_IXGRP) ? "x" : "-");
    strcat(combuf, (st.st_mode & S_IROTH) ? "r" : "-");
    strcat(combuf, (st.st_mode & S_IWOTH) ? "w" : "-");
    strcat(combuf, (st.st_mode & S_IXOTH) ? "x" : "-");
    strcat(combuf, " ");
    lto_u(buf, st.st_nlink, 3);
    strcat(combuf, buf);
    strcat(combuf, " ");
    lto_u(buf, st.st_uid, 3);
    strcat(combuf, buf);
    strcat(combuf, " ");
    lto_u(buf, st.st_gid, 3);
    strcat(combuf, buf);
    strcat(combuf, " ");
    lto_u(buf, (unsigned long)st.st_size, 10);
```

```
strcat(combuf, buf);
strcat(combuf, " ");
ctime_r(&st.st_mtime, buf);
strcat(combuf, buf);
```

2.3 ディレクトリ操作関数

mkdir　　ディレクトリの作成

形式　int mkdir(const char *dirname)

引数　dirname　　ディレクトリ名

戻値　0　　正常終了
　　-1　　エラー

説明　dirname で指定したディレクトリを作成します。ディレクトリ名の指定方法は、fopen におけるファイル名の指定方法と同様です。すでに存在するファイルやディレクトリと同じ名前を指定するとエラーになります。

例　ディレクトリ testdir を作成します。

```
TASK MainTask(void)
{
    int r;

    r = mkdir("C:¥testdir");
    if (r != 0)
        goto END;
```

rmdir　　ディレクトリの削除

形式　int rmdir(const char *dirname)

引数　dirname　　ディレクトリ名

戻値　0　　正常終了
　　-1　　エラー

説明　dirname で指定したディレクトリを削除します。ディレクトリ名の指定方法は mkdir と同様です。開いているディレクトリや存在しないディレクトリを指定するとエラーになります。

例　ディレクトリ testdir を削除します。

```
TASK MainTask(void)
{
    int r;

    r = rmdir("C:¥testdir");
    if (r != 0)
        goto END;
```

opendir　　ディレクトリのオープン

形式　DIR *opendir(const char *dirname)

引数　dirname　　ディレクトリ名

戻値　NULL 以外　　ディレクトリポインタ
　　NULL　　エラー

説明　dirname で指定したディレクトリを開きます。ディレクトリ名の指定方法は mkdir と同様です。共有ディレクトリを開く場合、C:¥.(ピリオド)のように指定します。正常終了すると readdir や closedir で指定するディレクトリポインタを返します。存在しないディレクトリを指定するとエラーになります。同時にオープンすることができるファイルとディレクトリ数の合計は nfscfg.h に定義されるマクロ NFS_NFILE の値で決定されます。既定値は 8 です。

例　ディレクトリ testdir を開きます。

```
TASK MainTask(void)
{
    DIR *dp;

    dp = opendir("C:¥testdir");
    if (!dp)
        goto END;
```

readdir　　ディレクトリエントリ情報の読み出し

形式　struct dirent *readdir(DIR *dp)

引数　dp　　ディレクトリポインタ

戻値　NULL 以外　　正常終了
　　NULL　　エントリが無いまたはエラー

説明　dp で指定したディレクトリからディレクトリエントリの情報を 1 個読み出し、結果を格納した dirent 構造体へのポインタを返します。最後のディレクトリエントリに達した場合、または、エラーの場合に NULL を返します。readdir が返すディレクトリエントリの情報は、同じディレクトリポインタ dp に対する readdir の呼び出しで上書きされます。dirent 構造体は nofnfs.h に以下のように定義されています。

```
#define NAME_MAX        256     エントリ名長さの最大値
struct dirent {
    unsigned long d_ino;        i-node 番号
    unsigned short d_namlen;    エントリ名の長さ
    char d_name[NAME_MAX+1];    エントリ名
};
```

例　ディレクトリ testdir から連続してディレクトリエントリ情報を読み出します。

```
TASK MainTask(void)
{
    DIR *dp;
    struct dirent *de;

    dp = opendir("C:¥testdir");
    if (!dp)
        goto END;
```

```
for (;;) {
    de = readdir(dp);
    if (!de)
        break;
    puts(de->d_name);
}
closedir(dp);
```

closedir　　ディレクトリのクローズ

形式　int closedir(DIR *dp)

引数　dp　　ディレクトリポインタ

戻値　0　　正常終了
　　-1　　エラー

説明　dp で指定したディレクトリを閉じます。

例　readdir の例を参照してください。

2.4 時間関数

localtime　　時間値を現地時間に変換

形式　struct tm *localtime(const time_t *t)
　　struct tm *localtime_r(const time_t *t, struct tm *ptm)

引数　t　　時間値へのポインタ
　　ptm　　変換結果を格納する tm 構造体へのポインタ

戻値　NULL 以外　　変換結果を格納した tm 構造体へのポインタ
　　NULL　　*t の値が 1970 年 1 月 1 日午前 0 時 0 分 0 秒以前

説明　t で指定した time_t 型の時間値、すなわち万国標準時(UTC)1970 年 1 月 1 日午前 0 時 0 分 0 秒からの秒単位の経過時間を現地時間(東京=UTC+9 時間)に変換し、その結果を格納した tm 構造体へのポインタを返します。tm 構造体は nofnfs.h に以下のように定義されています。

```
struct tm {
    int tm_sec;         秒(0-59)
    int tm_min;         分(0-59)
    int tm_hour;        時間(0-23)
    int tm_mday;        日(1-31)
    int tm_mon;         月(0-11、0 が 1 月)
    int tm_year;        現在の年から 1900 を引いた値
    int tm_wday;        曜日(0-6、0 が日曜日)
    int tm_yday;        その年の最初からの日数(0-365)
    int tm_isdst;       常に 0
};
```

localtime は変換結果を格納する tm 構造体を静的変数として内部に持つ非リエントラント(タスク間で排他的に使用できない)関数です。複数のタスクで localtime を使用する場合は、リエントラント関数 localtime_r を使用してください。

asctime　　tm 構造体を文字列に変換

形式　char *asctime(const struct tm *ptm)
　　char *asctime_r(const struct tm *ptm, char *buf)

引数

	ptm	tm 構造体へのポインタ
	buf	変換結果を格納するバッファへのポインタ

戻値

	NULL 以外	変換結果を格納したバッファへのポインタ
	NULL	ptm または buf が NULL

説明　ptm で指定した tm 構造体を以下の形式の文字列に変換し、その結果を格納したバッファへのポインタを返します。

Mon Jun 21 10:34:23 2004¥n　　(¥n と¥0 を含む文字列長は 26 バイト)

asctime は変換結果を格納するバッファを静的変数として内部に持つ非リエントラント(タスク間で排他的に使用できない)関数です。複数のタスクで asctime を使用する場合は、リエントラント関数 asctime_r を使用してください。

ctime　　時間値を文字列に変換

形式　char *ctime(const time_t *t)
　　char *ctime_r(const time_t *t, char *buf)

引数

	t	時間値へのポインタ
	buf	変換結果を格納するバッファへのポインタ

戻値

	NULL 以外	変換結果を格納したバッファへのポインタ
	NULL	t または buf が NULL

説明　t で指定した time_t 型の時間値を以下のような文字列に変換し、その結果を格納したバッファへのポインタを返します。

Mon Jun 21 10:34:23 2004¥n　　(¥n と¥0 を含む文字列長は 26 バイト)

ctime は変換結果を格納するバッファを静的変数として内部に持つ非リエントラント(タスク間で排他的に使用できない)関数です。複数のタスクで ctime を使用する場合は、リエントラント関数 ctime_r を使用してください。

第 3 章 エラーコード

NFS for NORTi のエラーコードは以下の通りです。

エラーコード	値	説明
EV_FNAME	-97	指定ファイル名、指定ディレクトリ名が異常
EV_FSAME	-98	同一ファイル名、同一ディレクトリ名が存在する
EV_NOFILE	-99	指定ファイルが存在しない
EV_NODIR	-100	指定ディレクトリが存在しない
EV_FMODE	-101	指定モードが異常
EV_NOOPEN	-102	ファイルがオープンされていない
EV_OPENED	-103	指定ファイルが既にオープンされている
EV_DISKRD	-104	デバイスリードエラー
EV_DISKWR	-105	デバイスライトエラー
EV_NFILE	-106	可能な同時オープンファイル数を超えた
EV_DISKFULL	-107	ディスクフル
EV_DRVNAME	-108	指定ドライブ名が異常
EV_FPAR	-109	指定パラメータが異常
EV_EOF	-110	ファイルの終端に達した
EV_DIRENT	-111	可能な同時オープンディレクトリ数を超えた
EV_FNOSPT	-112	未サポート
EV_DISKINI	-113	初期化(disk_ini)されていない
EV_FILEINI	-114	初期化(fsys_ini)されていない
EV_UNMOUNT	-115	マウントされていない
EV_NOEMPTY	-116	空でないディレクトリを削除しようとした
EV_DRVINI	-117	デバイス初期化エラー
EV_MOUNT	-118	デバイスマウントエラー
EV_CACHESZ	-119	キャッシュサイズ不足
EV_FATS	-120	FAT エラー
EV_NFSPAR	-121	NFS プロシジャパラメータエラー
EV_NFS	-122	NFS プロトコルエラー
EV_MOUNTP	-123	MOUNT プロトコルエラー
EV_PCNFS	-124	PCNFS プロトコルエラー
EV_PMAP	-125	Portmap プロトコルエラー
EV_RPC	-126	RPC プロトコルエラー
EV_XDR	-127	XDR エンコード/デコードエラー
EV_FERR	-128	その他のエラー

※ エラーコードは NORTi File System Version 4 と共通の為、NFS for NORTi では未使用のエラーコードも含まれています。

第4章 コンフィグレーション

4.1 マクロ

NFS for NORTi の通信処理部分は独立したタスクになっています。ファイルシステムタスクは、メールボックスを通じてアプリケーションから受信したファイル操作要求を順次処理します。ファイルシステムタスクのプライオリティやスタックサイズ、同時オープン可能なファイル数とディレクトリ数の合計、通信プロトコル、NFS サーバーに作成するファイルの保護モード、各種タイムアウト値、リトライ数はコンフィグレーションヘッダファイル nfscfg.h に定義されています。各マクロの既定値と意味は次の通りです。

マクロ名	既定値	説明
NFS_TSK_PRI	0	ファイルシステムタスク プライオリティ
NFS_TSK_SSZ	2048	ファイルシステムタスク スタックサイズ
NFS_NFILE	8	同時オープン可能なファイル数とディレクトリ数の合計
NFS_PROT	NFS_PROT_TCP	通信プロトコル
NFS_AMODE	00777	NFS サーバーに作成するファイルの保護モード(8 進数)
NFS_TMOUT	(5000/MSEC)	送受信タイムアウト値
NFS_CON_RETRY	10	TCP 接続リトライ数
NFS_UDP_TMOUT	(1000/MSEC)	UDP 受信タイムアウト初期値
NFS_UDP_RETRY	5	UDP 再送数
NFS_RPC_RETRY	5	RPC 要求再送数
NFS_TCP_PORT	1070	TCP 自分側ポート番号

これらの値は、アプリケーションのコンフィグレーションファイルで#include ″nfscfg.h″ の上位にマクロを定義することにより、ユーザーがカスタマイズすることができます。

例)nfscfg.c

```
#define NFS_NFILE       10              同時オープン可能なファイル数を 10 に設定
#define NFS_PROT        NFS_PROT_UDP    通信プロトコルに UDP を選択

#include ″nfscfg.h″
```

※NFS_TSK_PRI の既定値は 0 です。この場合、ファイルシステムタスクのプライオリティは nfs_ini を実行したアプリケーションタスクと同一の値に設定されます。NFS_TSK_PRI を 0 以外に設定する場合、TCP/IP プロトコルスタック IP タスクのプライオリティ(既定値 4)より大きい値、すなわち低い優先度を設定してください。IP タスクより高い優先度に設定すると自動的に(IP タスクの優先度＋１)の値が設定されます。

※NFS_PROT には通信プロトコルに TCP(NFS_PROT_TCP)と UDP(NFS_PROT_UDP)のどちらを使用するかを設定してください。

※NFS_AMODE には NFS サーバー上に作成するファイル(ディレクトリ)の保護モードを８進数で設定してください。各ビットの意味は関数 stat の st_mode の説明を参照してください。

※NFS_TCP_PORT には TCP 接続の自分側ポート番号を設定してください。TCP_PORTANY(0)を設定すると、TCP/IP プロトコルスタックが自動的に割り当てます。

4.2 リソース

NFS for NORTi が使用する各種リソースの数は、コンフィグレーションヘッダファイル nfscfg.h に NFS_N...で始まるマクロ名で定義されています。nfscfg.h を#include したアプリケーションのコンフィグレーションファイルにおいて、NFS_N...をアプリケーション全体で使用するリソースの数に加算してください。

例)nfscfg.c

```
#include "nfscfg.h"

#define TCP_REPID_MAX    6+NFS_NTCP_REP          TCP 受付口最大個数
#define TCP_CEPID_MAX    8+NFS_NTCP_CEP          TCP 通信端点最大個数
#define UDP_CEPID_MAX    3+NFS_NUDP_CEP          UDP 通信端点最大個数

#include "nonetc.h"

#define TSKID_MAX        5+TCP_NTSK+NFS_NTSK     タスク使用数
#define SEMID_MAX        1+TCP_NSEM+NFS_NSEM     セマフォ使用数
#define FLGID_MAX        1+TCP_NFLG+NFS_NFLG     イベントフラグ使用数
#define MBXID_MAX        1+TCP_NMBX+NFS_NMBX     メールボックス使用数
#define MBFID_MAX        4+TCP_NMBF+NFS_NMBF     メッセージバッファ使用数
#define PORID_MAX        1+TCP_NPOR+NFS_NPOR     ランデブ用ポート使用数
#define MPLID_MAX        1+TCP_NMPL+NFS_NMPL     可変長メモリプール使用数
#define MPFID_MAX        1+TCP_NMPF+NFS_NMPF     固定長メモリプール使用数
#define DTQID_MAX        1+TCP_NDTQ+NFS_NDTQ     データキュー使用数
#define MTXID_MAX        1+TCP_NMTX+NFS_NMTX     ミューテックス使用数
#define ISRID_MAX        1+TCP_NISR+NFS_NISR     割込みサービスルーチン使用数
#define CYCNO_MAX        2+TCP_NCYC+NFS_NCYC     周期ハンドラ使用数
#define ALMNO_MAX        1+TCP_NALM+NFS_NALM     アラームハンドラ使用数

#include "nocfg4.h"
```

※　上記の例で、6,8,3 や 5,1,1,1,4,…はアプリケーションが使用するリソースの数です。

第 5 章 ローカルファイルシステムとの併用

5.1 対応ローカルファイルシステム

NFS for NORTi と併用することができるローカルファイルシステムは NORTi File System Version 4 のみです。他のローカルファイルシステムと併用することは出来ません。

5.2 実装上の注意

NFS for NORTi と NORTi File System Version 4 を併用する場合、以下の点に注意して実装してください。

ヘッダファイル

アプリケーションのソースファイルに両ファイルシステムのヘッダファイルを#include する場合、必ず NFS for NORTi のヘッダファイル nofnfs.h を NORTi File System Version 4 のヘッダファイル nofile.h の下に#includcde してください。

例

```
#include “nofile.h”          /* NORTi File System Version 4 */
#include “nofnfs.h”          /* NFS for NORTi */
```

ドライブ名による識別と初期化

fopen などでファイル名を指定すると、ファイルシステムはパスの中のドライブ名によってファイル操作を行うファイルシステムを決定します。この判定を正しく行う為、初期化は NORTi File System Version 4 → NFS for NORTi の順に実行してください。NORTi File System Version 4 のディスクドライバ初期化関数 disk_ini で割り当てたドライブ名を、mount の引数 drive に指定するとエラーになります。

例　NORTi File System Version 4 の初期化、Comapct Flash ドライバ、RAM ディスクドライバの初期化、NFS for NORTi の初期化、mount の順に実行します。CompactFlash を B:に、RAM ディスクを A:に、NFS サーバーの共有ディレクトリを C:に割り当てます。

```
TASK MainTask(void)
{
```

```
ER ercd;

/* IP & Ethernet Address の読み出し */
read_ip_addr();
read_ethernet_addr();

/* TCP/IP プロトコルスタック初期化 */
ercd = tcp_ini();
if (ercd < 0)
    goto END;

/* NORTi File System Version 4 初期化 */
ercd = file_ini(file, LFS_NFILE, 0, 0);
if (ercd != E_OK)
    goto END;

/* CompactFlash ドライバ初期化 */
ercd = disk_ini(&disk[0], "B:", flash_ATA, 0, 0, NULL, 0, 0);
if (ercd != E_OK)
    goto END;
ercd = disk_cache(&disk[0], cache, NCACHE);
if (ercd != E_OK)
    goto END;

/* RAM ディスクドライバ初期化 */
ercd = disk_ini(&disk[1], "A:", ramdisk, DSK_ADDR, DSK_SIZE, NULL, 0, 0);
if (ercd != E_OK) {
    ercd = dformat("A:", 0);
    if (ercd != 0)
        goto END;
}

/* NFS for NORTi 初期化 */
ercd = nfs_ini();
if (ercd != E_OK)
    goto END;
```

```
/* マウント */
ercd = mount("pcnfslo", "pcnfspw", "192.168.0.11", "/TEMP", "C:", 0);
if (ercd != E_OK)
    goto END;
        :
```

5.3 動作の違い

ファイル名の(ドライブ名):¥を省略すると、NORTi File System Version 4 の disk_ini で最初に割り当てたドライブに対してファイル操作が行われます。NFS サーバー上のファイルを操作するには、必ず(ドライブ名):¥を付加してください。操作対象がディレクトリの場合も同様です。

例　　5.2 の初期化後、CompactFlash 上のファイル test.txt を開き、NFS サーバー上のファイル test.txt を削除します。

```
TASK MainTask(void)
{
    FILE *fp;
    int r;

    fp = fopen("test.txt", "r");
    if (!fp)
        goto END;

    r = remove("C:¥test.txt");
    if (r != 0)
        goto END;
            :
```

第 6 章 サンプルプログラム

6.1 ファイル構成

nfs*XXXX*.c
NFS for NORTi サンプルプログラムのメインソースファイルです。*XXXX* の部分はターゲットボードに依存します。NORTi File System Version 4 と NFS for NORTi を同時、またはそれぞれ単独で使用することができます。PC 上で動作するターミナルソフトを使用して、シリアルまたは TELNET 経由でターゲットにログインすることにより、NFS for NORTi がサポートする関数の動作を確認することができます。また、コンパイルオプションに以下のマクロを定義することにより、サンプルプログラムをカスタマイズできます。

マクロ名	既定値	説明
LITTLE_ENDIAN	(未定義)	定義 = リトルエンディアン、未定義 =ビッグエンディアン
DHCP	(未定義)	定義 = DHCP サーバーより動的に IP アドレス等を取得
CF	0	0 = CompactFlash 未使用、1 = CompactFlash を使用
NCACHE	10	CompactFlash 用キャッシュバッファ個数
FSOPTION	2	0 = NORTi File System Version 4 のみ使用 1 = NFS for NORTi のみ使用 2 = 両方とも使用

nfscfg.c
NFS for NORTi サンプルプログラムのコンフィグレーションファイルです。nfs*XXXX*.c と同様に、コンパイルオプションにマクロ FSOPTION を定義できます。

その他
サンプルプログラムをビルドして実行する為には、上記ファイルの他にターゲット用のシリアルドライバ、LAN ドライバ、NORTi¥SRC にあるシリアル入出力関数、NORTi¥NETSMP にある以下のソースファイルと対応するヘッダファイルが必要です。

NORTi¥SRC¥nosio.c	シリアル入出力関数
NORTi¥NETSMP¥SRC¥noncons.c	コンソール入出力関数
NORTi¥NETSMP¥SRC¥nondhcp.c	DHCP クライアント
NORTi¥NETSMP¥SRC¥nonedns.c	DNS リゾルバ
NORTi¥NETSMP¥SRC¥nonshel.c	コマンドシェル
NORTi¥NETSMP¥SRC¥nonteld.c	TELNET サーバー

6.2 起動

ここではNFS for NORTiに付属するサンプルプログラムの使用方法について説明します。

シリアル経由でターゲットにログインする為には、PCとNFS for NORTiが動作するターゲットをRS-232Cクロスケーブルで接続し、PCのターミナルソフトを以下の設定で起動してください。

ボーレート　　38400bps
キャラクタ長　8
パリティビット　無し
ストップビット　1
フロー制御　　無し

サンプルプログラムnfs*XXXX*.cのメインタスク(MainTask)では、TCP/IPの初期化、シリアルとTELNETコンソールの初期化、TELNETサーバーの初期化に続いて、NORTi File System Version 4の初期化、ディスクドライバの初期化、NFS for NORTiの初期化を行います。

サンプルプログラムを実行すると、ターミナルソフト上にターゲットの各種IPアドレスが表示されます。(DHCP使用時)

```
*** Network File System Sample Program ***
Success get data from DHCP server.
[Ethernet Address   ] : [**-**-**-**-**-**]
[Default IP Address ] : [192.168.0.31]
[Default Gateway    ] : [192.168.0.1]
[Subnet Mask        ] : [255.255.255.0]
[DNS IP Address     ] : [192.168.0.1]
[DHCP IP Address    ] : [192.168.0.1]
```

続いてコンソールサーバーからの出力として「login:」プロンプトが表示されます。ログイン名とパスワードは登録されていませんので、リターンキーのみでログインできます。

```
login:          (リターンキー)
Password:       (リターンキー)
>
```

「?」または「help」を入力してください。実行可能なコマンドの一覧が、パラメータの説明とともに表示されます。

```
>?
       ip
    mount [user name][password][IP address][directory name][drive name]
  unmount [drive name]
    fopen [file name][mode]
   fclose [file number]
    fgetc [file number]
    fputc [character][file number]
    fgets [file number]
    fputs [string][file number]
    fread [size][count][file number]
   fwrite [size][count][file number]
    ftell [file number]
    fseek [file number][offset][origin(0:SEEK_SET 1:SEEK_CUR 2:SEEK_END)]
   fflush [file number]
     feof [file number]
   ferror [file number]
 clearerr [file number]
   remove [file name]
   rename [old name][new name]
     stat [file name]
    mkdir [directory name]
    rmdir [directory name]
  opendir [directory name]
  readdir [directory number]
 closedir [directory number]
    speed [read/write size(1:1Mbyte 2:10Mbyte 3:100Mbyte)][drive name]
```

※コマンドと各パラメータの間は、スペースで区切って入力してください。

6.3 コマンド

以下にコマンドの使用方法を説明します。

ip

形式　ip

説明　MAC アドレスと各種 IP アドレスを表示します。

例　>ip

```
[Ethernet Address   ] : [**-**-**-**-**-**]     MAC アドレス
[Default IP Address ] : [192.168.0.31]          ターゲットの IP アドレス
[Default Gateway    ] : [192.168.0.1]           ゲートウェイ
[Subnet Mask        ] : [255.255.255.0]         サブネットマスク
```

mount

形式　mount [ユーザー名][パスワード][IP アドレス][ディレクトリ名][ドライブ名]

説明　ユーザー名、パスワードを使用して認証を行い、IP アドレスの NFS サーバー上の共有ディレクトリをマウント、ドライブ名に割り当てます。

例　ユーザー名 pcnfslo、パスワード pcnfspw を使用して、IP アドレスが 192.168.0.92 の NFS サーバー上の共有ディレクトリ TEMP をドライブ C:にマウントします。

```
>mount pcnfslo pcnfspw 192.168.0.92 /TEMP C:
return value = 0
```

unmount

形式　unmount［ドライブ名］

説明　mount コマンドでマウントしたドライブをアンマウントします。

例　ドライブ C:をアンマウントします。

```
>unmount C:
return value = 0
```

fopen

形式　fopen［ファイル名][オープンモード]

説明　指定したファイル名とオープンモードでファイルを開きます。正常に終了すると、戻り値とファイルに割り当てられたファイル No.が表示されます。

例　ファイル C:¥test.txt を書き込みモードで開きます。

```
>fopen C:¥test.txt w
return value = 0x0042BE90
file number = 0
```

fclose

形式　fclose［ファイルNo.］

説明　指定したファイルを閉じます。

例　ファイルNo.0のファイルを閉じます。

```
>fclose 0
return value = 0
```

fgetc

形式　fgetc［ファイルNo.］

説明　指定したファイルから1文字読み出します。

例　ファイルNo.0のファイルから'a'を読み出します。

```
>fgetc 0
return value = 97
```

fputc

形式　fputc［書込む文字][ファイルNo.］

説明　指定したファイルに1文字書き込みます。

例　ファイルNo.0のファイルに'a'を書き込みます。

```
>fputc a 0
return value = 97
```

fgets

形式　fgets［ファイルNo.］

説明　指定したファイルからバッファに文字列を読み出します。

例　ファイルNo.0のファイルから文字列を読み出します。

```
>fgets 0
abcdefghijklmn
return value = 0x8C016868
```

fputs

形式　fputs［文字列][ファイルNo.］

説明　指定したファイルに文字列を書き込みます。

例　ファイルNo.0のファイルに文字列"abcdefg"を書き込みます。

```
>fputs abcdefg 0
return value = 0
```

fread

形式　fread［バイト単位の項目サイズ][読み出す最大項目数][ファイルNo.］

説明　指定したファイルからバッファにデータを読み出します。

例　ファイルNo.0のファイルから1024バイト読み出します。

```
>fread 1 1024 0
return value = 1024
```

fwrite

形式　fwrite［バイト単位の項目サイズ][書き込まれる最大項目数][ファイルNo.]

説明　指定したファイルにバッファのデータを書き込みます。

例　ファイルNo.0のファイルに1024バイト書き込みます。

```
>fwrite 1 1024 0
return value = 1024
```

ftell

形式　ftell［ファイルNo.]

説明　指定したファイルの現在の読み書き位置を取得します。

例　ファイルNo.0のファイルの現在の読み書き位置は、先頭から14バイト目です。

```
>ftell 0
return value = 14
```

fseek

形式　fseek［ファイルNo.][初期位置からのバイト数][初期位置]

説明　指定した位置にファイルの読み書き位置を移動します。初期位置には、以下の３つの値のいずれかを指定します。

0 ： ファイルの先頭
1 ： 現在位置
2 ： ファイルの終端

例　ファイルNo.0のファイルの読み書き位置を先頭に移動します。

```
>fseek 0 0 0
return value = 0
```

fflush

形式　fflush［ファイルNo.]

説明　指定したファイルのREAD/WRITEバッファの内容をフラッシュします。

例　ファイルNo.0のファイルのREAD/WRITEバッファをフラッシュします。

```
>fflush 0
return value = 0
```

feof

形式　feof［ファイルNo.］

説明　指定したファイルの現在の読み書き位置がファイル終端かどうか調べます。

例　ファイルNo.0のファイルの終端チェックを行います。

```
>feof 0
return value = 0
```

ferror

形式　ferror［ファイルNo.］

説明　指定したファイルのエラーコードを表示します。

例　ファイル No.0 のファイルのエラーコードを表示します。エラーコードはEV_FPAR(-109)です。

```
>ferror 0
return value = -109
```

clearerr

形式　clearerr［ファイルNo.］

説明　指定したファイルのエラーコードとEOFフラグをクリアします。

例　ファイルNo.0のファイルのエラーコードとEOFフラグをクリアします。戻値はありません。

```
clearerr 0
```

remove

形式　remove［ファイル名］

説明　指定したファイルを削除します。

例　ファイルC:¥test.txtを削除します。

```
>remove C:¥test.txt
return value = 0
```

rename

形式　rename［古いファイル名][新しいファイル名］

説明　古いファイル名で指定したファイルの名前を、新しいファイル名に変更します。

例　ファイルC:¥test.txtのファイル名を、sss.txtに変更します。

```
>remove C:¥test.txt sss.txt
return value = 0
```

stat

形式　stat［ファイル名］

説明　指定したファイルの状態情報を表示します。

例　ファイル C:¥test.txt の状態情報を表示します。

```
>stat C:¥test.txt
-rwxrwxrwx   1 100 100     234520 Fri May 14 00:47:58 2004
```

mkdir

形式　mkdir［ディレクトリ名］

説明　指定したディレクトリを作成します。

例　ディレクトリ C:¥testdir を作成します。

```
>mkdir C:¥testdir
return value = 0
```

rmdir

形式　rmdir［ディレクトリ名］

説明　指定したディレクトリを削除します。

例　ディレクトリ C:¥testdir を削除します。

```
>rmdir C:¥testdir
return value = 0
```

opendir

形式　opendir［ディレクトリ名］

説明　指定したディレクトリを開きます。正常終了すると、戻値とディレクトリに割り当てられたディレクトリ No.を表示します。

例　ディレクトリ C:¥testdir を開きます。

```
>opendir C:¥testdir
return value = 0x8C01E8F0
directory number = 0
```

readdir

形式　readdir［ディレクトリ No.］

説明　指定したディレクトリのディレクトリエントリ情報を全て読み出し、i-node 番号、エントリ名長さ、エントリ名の順に表示します。

例　ディレクトリ No.0 のディレクトリの、ディレクトリエントリ情報を全て読み出します。

```
>readdir 0
0x00677C01   1 .
0x00000505   2 ..
0x00098A32   8 test.txt
return value = 0x00000000
```

closedir

形式　closedir［ディレクトリ No.］

説明　指定したディレクトリを閉じます。

例　ディレクトリ No.0 のディレクトリを閉じます。

```
>closedir 0
return value = 0
```

speed

形式　speed [データサイズ(1:1M バイト 2:10M バイト 3:100M バイト)][ドライブ名]

説明　指定したドライブ上に(クライアントの IP アドレス).txt を作成し、指定したサイズの書き込みと読み込みを実行します。また、書き込みと読み込みに要した時間をそれぞれ測定して表示します。

例　ドライブ C:に 10M バイトのデータの書き込みと読み込みを実行し、速度を測定します。

```
>speed 2 C:
fopen c:¥192.168.0.5.txt w
fwrite ...
fclose
size  =   10485760byte
time  =      10220msec
speed =       8208kbps
fopen c:¥192.168.0.5.txt r
fread ...
fclose
size  =   10485760byte
time  =       6990msec
speed =      12000kbps
```

連続実行

コマンド fgetc、fputc、fgets、fputs、fread、fwrite、readdir は、最後に実行回数を指定することにより、連続して実行することができます。

例　　ファイル No.0 のファイルに対して、fgetc を 10 回実行します。

```
>fgetc 0 10
return value = 13
return value = 10
return value = 129
return value = 64
return value = 129
return value = 64
return value = 129
return value = 64
return value = 129
return value = 64
```

第7章 Microsoft Windows Services for UNIX 3.5

ここでは NFS サーバーとして使用する Microsoft Windows Services for UNIX 3.5(以下 SFU3.5)のインストール方法と設定方法について説明します。

SFU3.5 は 2006 年 1 月現在、マイクロソフト株式会社より無償ダウンロードできます。以下の SFU3.5 ホームページより SFU3.5 日本語版をダウンロードしてください。
http://www.microsoft.com/japan/windows/sfu/
上記ホームページには SFU3.5 を使用する上で必要とされるシステムの最低要件やライセンスなどの重要な情報が掲載されています。これらも必ずご一読ください。

SFU3.5 がインストールできる以下の OS の中で、1)と 3)はサーバー製品の為、NFS クライアントにはクライアント・アクセス・ライセンス(CAL)が必要です。また、2)と 4)はワークステーション製品の為、同時接続できるクライアント数に制限がありますのでご注意ください。

1) Microsoft Windows Server 2003
2) Service Pack 1 が適用されている Windows XP Professional
3) Service Pack 3 以降が適用されている Windows 2000 Server
4) Service Pack 3 以降が適用されている Windows 2000 Professional

クライアント・アクセス・ライセンス(CAL)
サーバーOS やクライアント OS のライセンスとは別に、クライアントがサーバーへアクセスすることを許諾するライセンスです。マイクロソフトのサーバー製品では、サーバーにアクセスするためにアクセスするユーザーまたはデバイスごとに CAL を取得する必要があります。デバイスとは、デスクトップ PC、ノート PC、PDA など、サーバーにアクセスする端末を指します。

参考 URL
マイクロソフト Licensing ホーム
http://www.microsoft.com/japan/licensing/

7.1 インストール

①ダウンロードしたファイルを展開して、setup.exe を実行してください。

②以下の画面が順番に表示されます。必要な情報を入力して[次へ]をクリックしてください。

・Microsoft Windows Services for UNIX セットアップウィザードの開始

・ユーザー情報

・ライセンスとサポート情報

③[インストール オプション]画面が表示されます。[カスタム インストール]を選択して[次へ]をクリックしてください。

④[コンポーネントの選択]画面が表示されます。ツリー表示されるコンポーネントのアイコンの左に[+]マークがあるものは、マークをクリックするとサブコンポーネントが表示されます。アイコンの色が白いものはインストールするコンポーネントとして選択されています。[×]印がついているものは選択されていません。グレーのものはサブコンポーネントの一部が選択されています。コンポーネントのアイコンをクリックして[ローカルドライブにインストール]をクリックするとコンポーネントが選択されます。サブコンポーネントを持つコンポーネントは、[ローカルドライブにすべてインストール]をクリックすると全てのサブコンポーネントが選択されます。

以下のコンポーネントを選択して、[次へ]をクリックしてください。

・NFS → NFS サーバー

・NFS 認証ツール → ユーザー名マッピング

・NFS 認証ツール → NFS 認証サーバー

・NFS 認証ツール → PCNFS サーバー

⑤[セキュリティの設定]画面が表示されます。[次へ]をクリックしてください。

⑥[ユーザー名マッピング]画面が表示されます。[ローカル ユーザー名マッピングサーバー]→[パスワードファイルおよびグループファイル]を選択して、[次へ]をクリックしてください。

⑦パスワードとグループファイルの設定画面が表示されます。ここでは何も入力せずに[次へ]をクリックしてください。

⑧[インストール場所]画面が表示されます。[次へ]をクリックしてください。インストールが開始します。

⑨インストールを完了するために、PC の再起動を要求するメッセージが表示されます。[はい]をクリックして PC を再起動してください。インストールが正常に完了すると、以下の場所に[Services for UNIX の管理]が作成されます。

[スタート]→[プログラム]→[Windows Services for UNIX]→[Services for UNIX の管理]

7.2 NFS サーバーの設定

①[Services for UNIX の管理]を起動して、左側のツリービューで[NFS サーバー]をクリックしてください。
②右側の[ローカルコンピュータ上の NFS サーバー]で[クライアントグループ]タブをクリックしてください。
③[グループ名]にグループ名"nortigr"を入力して[新規]ボタンをクリックしてください。
④下にある[詳細情報]をクリックしてください。
⑤一番下のテキストボックスにグループに追加するクライアントの IP アドレスを入力し、[クライアントの追加]をクリックしてください。
⑥[クライアントの一覧 - nortigr:]に追加した IP アドレスが表示されることを確認してください。
⑦全てのクライアントについて⑤、⑥を繰り返してください。
⑧[サーバーの設定]タブをクリックして、[サーバーオプション]で[TCP サポートを有効にする]と[NFS V3 サポートを有効にする]のチェックが ON であることを確認してください。
⑨画面右上の[適用]ボタンをクリックしてください。

7.3 PCNFS サーバーの設定

①左側のツリービューで[PCNFS サーバー]を選択して、右側の[ローカル コンピュータ上の PCNFS サーバー]で[グループ]タブをクリックしてください。

②[グループ名]に"pcnfsgr"を入力して、[新規]をクリックしてください。

③"pcnfsgr"に GID 100 が自動的に割り当てられ、[現在のグループ:]に表示されることを確認してください。

④[ユーザー]タブを選択して、[新規]をクリックしてください。

⑤[PCNFS ユーザー -- Web ページ ダイアログ]が表示されますので、以下のように入力して[OK]をクリックしてください。

ユーザー名	pcnfsun
ユーザー ログオン名	pcnfslo
パスワード	pcnfspw
パスワードの確認	pcnfspw
プライマリグループ	pcnfsgr
ユーザーID	(入力しません)

※ここで設定したユーザーログオン名とパスワードを、NFS for NORTi のマウント関数 mount の引数 user と pass に指定します。

⑥[すべてのユーザー:]に入力したユーザーの情報と、自動的に割り当てられた UID 100 が表示されることを確認してください。

⑦[グループ]タブを再度選択して、[すべてのユーザー:]に新規作成したユーザー"pcnfslo"が表示されることを確認してください。

⑧一番下の[追加]をクリックして、[ユーザー - pcnfsgr:]にユーザーpcnfslo が表示されることを確認してください。

⑨画面右上の[適用]ボタンをクリックしてください。

7.4 ユーザー名マッピングの設定

①左側のツリービューで[ユーザー名マッピング]を選択して、右側の[ローカルコンピュータ上のユーザー名マッピング]で[構成]タブをクリックしてください。
②[パスワードファイルとグループファイルを使用する]をクリックしてください。
③[パスワードファイルのパスと名前]と[グループファイルのパスと名前]にそれぞれ以下のファイル名をフルパスで入力するか、[参照]をクリックしてファイルを選択してください。

パスワードファイル　　C:¥WINNT¥system32¥drivers¥etc¥passwd
グループファイル　　C:¥WINNT¥system32¥drivers¥etc¥group

④[マップ]タブを選択して、[グループマップの表示]をクリックしてください。
⑤[Windows グループの一覧]をクリックします。[Windows グループ:]に現在 Windows に登録されているローカルグループ名の一覧が表示されること確認してください。
⑥[Windows グループ:]で"Guests"を選択して、[Windows グループ名:]に"Guests"が表示されることを確認してください。
⑦[UNIX グループの一覧]をクリックします。[UNIX グループ:]に"pcnfsgr"が表示されることを確認してください。
⑧[UNIX グループ:]で"pcnfsgr"を選択して、[UNIX グループ名:]に"pcnfsgr"が表示されることを確認してください。
⑨[追加]をクリックしてください。[特別な Windows アカウントを指定しました。詳細マッピングをこのアカウントに設定しますか？]が表示されますので、[OK]をクリックしてください。
⑩[マップされたグループ:]で、Windows グループ"Guests"と UNIX グループ"pcnfsgr"が正しくマッピングされていることを確認してください。
⑪[ユーザー マップの表示]をクリックしてください。
⑫[Windows ユーザーの一覧]をクリックします。[Windows ユーザー:]に現在 Windows に登録されているローカルユーザー名の一覧が表示されること確認してください。
⑬[Windows ユーザー:]で"Guest"を選択して、[Windows ユーザー名:]に"Guest"が表示されることを確認してください。
⑭[UNIX ユーザーの一覧]をクリックします。[UNIX ユーザー:]に"pcnfslo"が表示されることを確認してください。
⑮[UNIX ユーザー:]で"pcnfslo"を選択して、[UNIX ユーザー名:]に"pcnfslo"が表示されることを確認してください。
⑯[追加]をクリックしてください。[特別な Windows アカウントを指定しました。詳細マッピングをこのアカウントに設定しますか？]が表示されますので、[OK]をクリックしてく

ださい。
⑰[マップされたユーザー:]で Windows ユーザー"Guest"と UNIX ユーザー"pcnfslo"が正しくマッピングされていることを確認してください。
⑱画面右上の[適用]ボタンをクリックしてください。

【重要】
クライアントをマッピングする Windows アカウントは Guest などのビルトインアカウントである必要はありません。クライアントをマッピングする前に、マッピング先の Windows アカウントが有効になっていることを必ず確認してください。Windows2000 Professional の場合は、[コントロールパネル]→[ユーザーとパスワード]→[詳細]タブ→[高度なユーザー管理]→[詳細]→[ローカルユーザーとグループ]の左側にツリー表示される[ユーザー]を選択して、右側のユーザー一覧で Guest を選択して右クリックし、[Guest のプロパティ]→[全般]タブで[アカウントを無効にする]のチェックが OFF である事を確認してください。

7.5 ディレクトリの共有

①NFS サーバー上の任意の場所にフォルダ"TEMP"を作成してください。
②エクスプローラ上でフォルダ TEMP を右クリックして[TEMP のプロパティ]を表示してください。
③[NFS 共有]タブを選択して、[このフォルダを共有する]をクリックしてください。
④[アクセス権]をクリックして[NFS 共有アクセス権]を表示し、[追加]をクリックして[クライアントとクライアントグループの追加]を表示してください。
⑤[名前]の一覧に表示される"nortigr"を選択して、[追加]→[OK]をクリックしてください。
⑥[NFS 共有アクセス権]と[TEMP のプロパティ]でそれぞれ[OK]をクリックしてください。

7.6 共有ディレクトリのマウント

ここまでの操作で、NFS サーバー上の共有ディレクトリ"TEMP"をマウントする為の準備が全て整いました。NORTi 上のアプリケーションで以下のようにマウントを実行してください。

例　　ユーザーログオン名が pcnfslo、パスワードが pcnfspw、NFS サーバーの IP アドレスが 192.168.0.11、共有ディレクトリ名が TEMP、ドライブ名が C:の場合

```
ercd = mount("pcnfslo", "pcnfspw", "192.168.0.11", "/TEMP", "C:", 0);
if (ercd != E_OK)
        :
```

第 8 章 その他

8.1 制限事項

・NFS for NORTi がサポートしていない ANSI C 関数をアプリケーションからコールすると標準ライブラリの関数がリンクされます。サポート外の関数は使用しないでください。
・複数の NFS クライアントによる NFS サーバー上の同一ファイルへの同時アクセスに対する排他制御は行っていません。アプリケーションまたはシステムで排他制御を行ってください。
・テキストモードは未サポートです。
・処理できるファイルサイズの上限は２ギガバイトです。
・同時にマウントできる共有ディレクトリは１つです。

8.2 トラブルシューティング

Q.マウントできない。
A.以下の点を確認してください。
・サーバーでファイヤウォール関連のソフトウェアが動作している場合は停止してください。
・クライアントとサーバーで、ログオン名とパスワードが正しく設定されていることを確認してください。
・クライアントの MAC アドレス/IP アドレスが正しく設定されていることを確認してください。
・mount に引数として与える共有ディレクトリ名の先頭に'/'(スラッシュ)が付いていることを確認してください。
・途中に半角スペースがある名前のディレクトリを共有すると、共有名のスペースは'_'(アンダーライン)に置き換えられます。スペースをアンダーラインに置き換えた名前でマウントしてください。
例)SHARE FOLDER → ″/SHARE_FOLDER″
・ユーザー名マッピングの設定で、クライアントのユーザーログオン名をマッピングした Windows アカウントが無効になっている場合、マウント処理は EV_RPC を返して失敗します。Windows アカウント Guest を有効にする場合は、[コントロールパネル]→[ユーザーとパスワード]→[詳細]タブ→[高度なユーザー管理]→[詳細]→[ローカルユーザーとグループ]の左側にツリー表示される[ユーザー]を選択し、右側のユーザー一覧で Guest を選択して右クリックし、[Guest のプロパティ]→[全般]タブで[アカウントを無効にする]

のチェックを OFF にしてください。

Q. 読み込みはできるが、書き込みができない。

A. クライアントの IP アドレスがクライアントグループに追加されており、クライアントグループのアクセス権が読み取り専用になっていないことを確認してください。共有ディレクトリのプロパティの[NFS 共有]→[アクセス権]→[名前]に現在そのディレクトリにアクセスが許可されているクライアントグループと ALL MACHINES が表示されます。ALL MACHINES の[アクセス権の種類]は既定で読み取り専用です。これは認証された全てのクライアントに読み取り専用のアクセス権が許可される事を意味します。セキュリティ上の問題が無い場合は、ALL MACHINES に読み込み書き込みのアクセス権を与えることもできます。

Q. NFS サーバーにクライアントが同時接続できない。

A. SFU3.5のクライアント同時接続数の最大値の既定値は16です。16台より多いクライアントを同時接続する場合は、NFSサーバー上のレジストリで、以下の場所にDWORD値レジストリRpcMaxConcurrentConnectionsPerIpを作成して値を定義し、クライアント同時接続の最大値を増加させてください。(SFU3.5ヘルプより)

HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Services for Unix¥Global

注)値は 16 と 64 の間でなければなりません。

NFS for NORTi ユーザーズガイド

株式会社ミスポ	http://www.mispo.co.jp/
一般的なお問い合わせ	sales@mispo.co.jp
技術サポートご依頼	norti@mispo.co.jp