第4章 TCP/IPネットワーク機能を使う

TCP/IPプロトコル・スタックTINETの実装

横田 敬久

TOPPERS/JSPには専用のTCP/IPプロトコル・スタックとしてTINETがある．この章では付属ColdFireマイコン基板向けにTINETの移植事例と活用方法を紹介する．（筆者）

1．TINETとは

TINETは，μITRONのTCP/IP準拠のAPIと拡張されたIPv6用のAPIを提供するTOPPERS/JSP用のプロトコル・スタックです．苫小牧工業高校専門学校の安部 司氏によって開発され，BSDベースのライセンスで配布されています．

TINETは**表1**のようなAPIを持っています（IPv4）．TINETはIPv6にも対応していますが，今回の移植ではIPv4環境を想定しているため，IPv6の実装は行っていません．

TINETでサポートしている「省コピーAPI」とは，tcp_snd_datのようにプログラムからTINETが使用しているTCPの送受信ウィンドウのバッファを直接得ることにより，プログラムのTCP/IPプロトコル・スタックへのデータ・コピーの手間を省略するAPIです．一般的なソケットAPIと比較して，データ・コピーが不要な分，高速に処理できます．

表1 TINETのAPI

TCP受付口の作成	
TCP_CRE_REP	TCP受付口の作成［静的API］
(tcp_cre_rep)	TCP受付口の作成（拡張機能）TINET 1.4から
(tcp_del_rep)	TCP受付口の作成（拡張機能）TINET 1.4から
TCP_CRE_CEP	TCP通信端点の作成［静的API］
TCP受付口・通信端点の予約	
VRID_TCP_REP	TCP受付口の予約ID （TINET独自）［静的API］
VRID_TCP_CEP	TCP通信端点の予約ID（TINET独自）［静的API］
オープン・クローズ	
tcp_acp_cep	接続要求待ち（受動オープン）
tcp_con_cep	接続要求（能動オープン）
tcp_sht_cep	データ送信の終了
tcp_cls_cep	通信端点のクローズ
送受信	
tcp_snd_dat	データ送信API
tcp_rcv_dat	データ受信API
省コピーAPI	
tcp_get_buf	送信用バッファの取得
tcp_snd_buf	バッファ内データの送信
tcp_rcv_buf	受信用バッファの取得
tcp_rel_buf	受信用バッファの開放
UDP通信端点	
UDP_CRE_CEP	UDP通信端点の生成［静的API］
(udp_cre_cep)	UDP通信端点の生成（拡張機能）TINET 1.4から
(udp_del_cep)	UDP通信端点の削除（拡張機能）TINET 1.4から
UDP送受信	
udp_snd_dat	データ送信API
udp_rcv_dat	データ受信API

2．付属ColdFireマイコン基板でのTINETの使用方法

● 付属ColdFireマイコン基板での注意

付属ColdFireマイコン基板のUDP対応GDBスタブを使用すると，GDBスタブによる通信でEthernet通信が占有されてしまいます．そのためTINETを使用してデバッグを行う際には，BDMを使用した開発か，シリアル版の

```
cd tinet/cfg
make
```

図1 コンフィグレータのメイク

```
cd sample1n

Makefileを編集
tinet_app_config.h を編集

make-3.79.1.exe depend
make-3.79.1.exe
```

図2 sample1nのメイク

KEYWORD ——TCP/IP，Ethernet，TINET，TOPPERS/JSP，省コピーAPI

GDBスタブを使用して開発を行う必要があります．デバッグ機能を犠牲にしてもよいなら，GDBスタブを使った後半128Kバイトへのロード・実行や，後半128Kバイトへのロード・実行もROMの容量が許す限り可能です．

● TINETを使ってみよう

現在リリースされているTINET 1.4を使用したアプリケーションの作成には，GNU make-3.79.1のインストールが必要です．現在のCygwinにはGNU make-3.8以降がインストールされているので，使用されている環境によってはメイク時にエラーが発生するため，別途GNU make-3.79.1をダウンロードして作成してください．

● TINETを使ったアプリケーションのメイク

TINETではμITRONのTCP/IPの静的APIを処理するために専用のコンフィグレータを使用するので，まずはこれをメイクします(**図1**)．

TINETのアプリケーションをconfigureで空のプロジェクトから作成するのは，厄介な作業になります．今回はサンプル・プログラムのメイクまでを行います(**図2**)．

sample1nは，TOPPERS/JSPのsample1のTelnet対応版です．TINETではアプリケーションの作成に**図3**のファイルを必要とします．

まずtinet_app_config.hは，ローカルIPや経路設定のマクロを設定します．route_cfg.cはtinet_app_config.hの設定を元にネットワーク上の経路情報を定義します．route_cfg.cは特殊な設定を行わなくてもechoサーバなどのサンプルを再利用すればほとんどの場合は間に合うと思います．

tinet_$(アプリケーション名).cfgにおけるアプリケーション名がsample1nならば，ファイル名はtinet_sample1n.cfgになります．このファイルはTINETライブラリ用の静的APIを定義するファイルです．ITRONのTCP/IPのAPIで定義されている端点情報や拡張されたIPv6用の静的APIをここで定義します．

例えば，sample1nのtinet_sample1n.cfgを見てみましょう(**リスト1**)．sample1nのtinet_sample1n.cfgを見ると，telnetプロトコルを実現するためにTCPの受付口と通信端点を定義しています．受付口のアドレス部にある{IPV4_ADDRANY, 23 }は23番ポートにアクセスしてきたすべてのアドレスに対して受け付けることを意味します．受付口で受け付けられた端点は，TCPの通信端点に接続して通信が成立します．

この場合，通信端点は一つしか定義されていないので，sample1nでは受け付けされた通信から同時に一つのTCP接続だけが成立します．二つ以上端点を受け付ける場合にはその数の分だけ端点を定義しタスクも割り付けてください．

minsvはTOPPERS/JSPで作ったWWWサーバとechoサーバが入っていて，ネットワーク統計情報を表示できます．minisvのメイク方法は**図4**のとおりです．

これらのアプリケーションのサーバ・クライアントの本体はtinet/netappにあり，各種サンプル・アプリケーションがモジュールとして組み込めるような形で提供されています．付属のサンプルであってもtinet_app_config.hのIPV4_ADDR_LOCALの部分などは書き換える必要があります．

3．デバイス・ドライバの作成

ここからはTINETのドライバについて解説します．

TINETのデバイス・ドライバとして用意されているの

リスト1
tinet_sample1n.cfg

```
/* TCP 受付口 */

TCP_CRE_REP (TCP_REPID, { 0,
{ IPV4_ADDRANY, 23 } } );

/* TCP 通信端点 */

TCP_CRE_CEP (TCP_CEPID, {
    0,
    tcp_swbuf,
    TCP_SWBUF_SIZE,
    tcp_rwbuf,
    TCP_RWBUF_SIZE,
    (FP)callback_nblk_tcp
    } );
```

```
tinet_app_config.h              TINETアプリケーション用のコンフィグレーション・ファイル
tinet_$(アプリケーション名).cfg  TINET用のコンフィグレータ用ファイル
route_cfg.c                     経路情報設定ファイル
```

図3　必要なファイル

```
cd minsv

Makefileを編集
tinet_app_config.h を編集

make-3.79.1.exe depend
make-3.79.1.exe
```

図4　minisvのメイク