ネットワークプログラミング講座
-UNIX, Windows & Java 環境における-

石井 秀治(NEC)
shuji@ccs.mt.nec.co.jp
日比野 洋克(オレンジソフト)
nonki@orangesoft.co.JP

目的

- ネットワークプログラムとはどのようなものであるかを理解する(初心者)
- WINDOWSのプログラム開発環境および，ネットワークプログラミングの基礎を理解する(UNIXプログラマ)
- UNIXのプログラム開発環境および，ネットワークプログラミングの基礎を理解する(WINDOWSプログラマ)

内容

- インターネット概要(石井)
- ネットワークプログラミング
  - UNIX編 (石井)
  - WINDOWS編(日比野)
  - JAVA編(日比野)
- 実例紹介(日比野･石井)
  - SMTP, POP
- まとめ

NEC

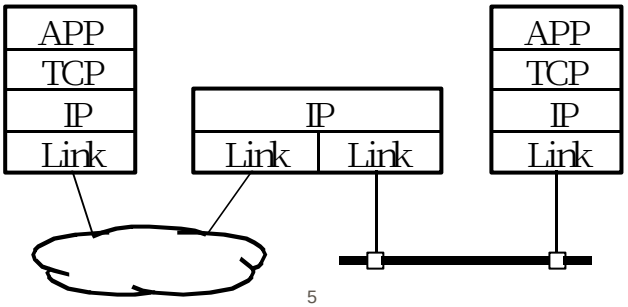
インターネット概要

- インターネット概要(石井)
- ネットワークプログラミング
  - UNIX編 (石井)
  - WINDOWS編(日比野)
  - JAVA編(日比野)
- 実例紹介(日比野･石井)
  - SMTP, POP
- まとめ

NEC

インターネット

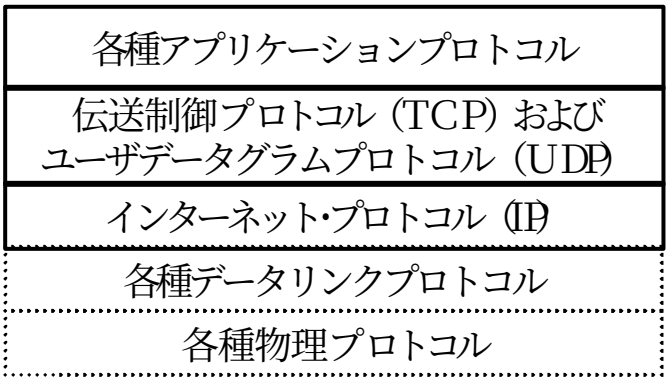
- 異なったメディアを統合して論理的なネットワークに見せる技術
  - ＩPアドレス
  - 経路制御



NEC

TCP/IP

- TCP, UDP とIPを中心にしたプロトコルスタック



NEC

IPアドレスとホスト名

- IP アドレス: 固定長の論理アドレス
  - IPv4: 32ビット, IPv6: 128ビット
  - ホストのネットワークインターフェースを識別
  - “192.168.0.3”, “3ffe::501:0:1001::2”
- ホスト名: ホストを識別する名前
  - 機械, プログラムはアドレス(数字)を好む
  - ユーザ(人間)は名前を好む
  - www.nic.ad.jp

トランスポート

- IPではデータ配送に信頼性がない
  - パケットが届かないかもしれない
  - 送信した順序と違う順序で受信するかもしれない
  - パケットが重複して到着するかもしれない
- 上位層(トランスポート)で信頼性を確保
  - TCP
  - UDP

TCP

- 信頼性のあるデータ配送
  - ストリーム指向
  - バーチャルサーキット(コネクション型)
  - バッファ付き転送
  - 全二重
- ユーザが下位プロトコルを意識せずに通信を行う

UDP

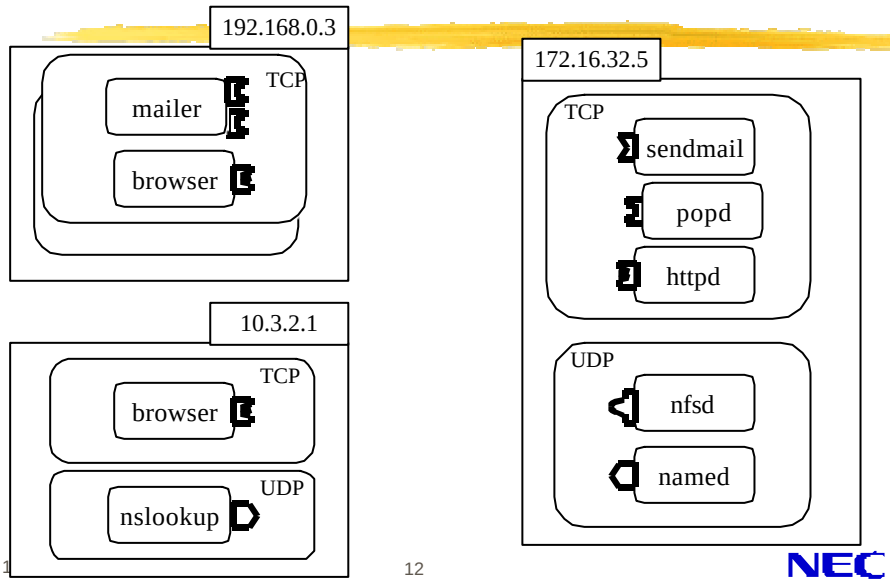
- 信頼性の無いデータグラム配送
  - = IPパケット+ポート番号+チェックサム
- ユーザ(プログラマ) にプロトコルを開放
  - NFS
  - TFTP
  - DHCP
  - DNS
  - …

ポート

- ホスト内での通信端点
  - 同一ホスト内のサービスを区別
- ポート番号
  - ポートを区別するための16ビットの正の整数
- well-known port
  - 特定のサービスのために予約されたポート
- トランスポートごとに独立
  - TCP の80番と UDPの80番は異なる

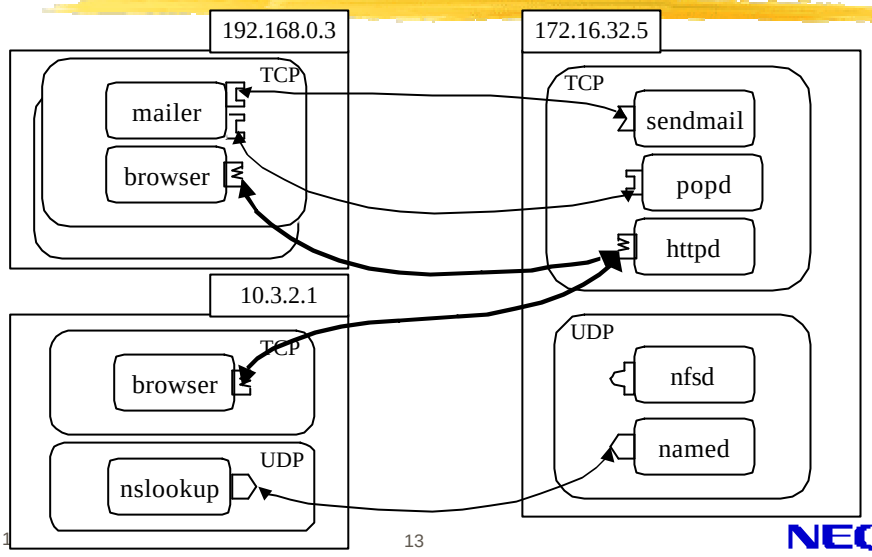
NEC

ポート(2)



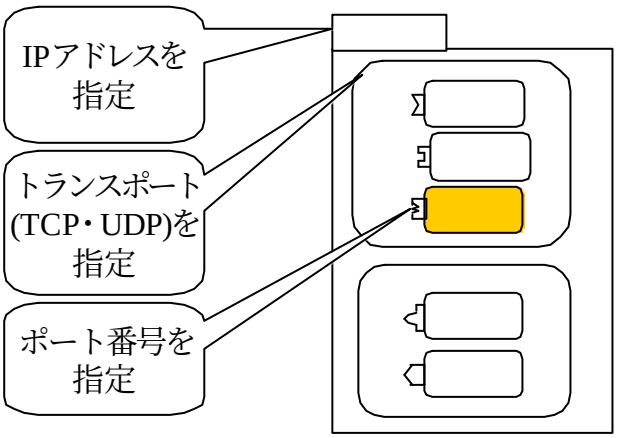
NEC

ポート(3)



NEC

通信端点の指定方法



NEC

プロセス間通信 API

- Berkeley socket
- TLI/XLI
- sunrpc
- winsock
- RMI
- WinInet
- :

NEC

ネットワークプログラミング

- インターネット概要(石井)
- ネットワークプログラミング
  - UNIX編 (石井)
  - WINDOWS編(日比野)
  - JAVA編(日比野)
- 実例紹介(日比野･石井)
  - SMTP, POP
- まとめ

UNIX 編

- インターネット概要(石井)
- ネットワークプログラミング
  - UNIX編 (石井)
  - WINDOWS編(日比野)
  - JAVA編(日比野)
- 実例紹介(日比野･石井)
  - SMTP, POP
- まとめ

- socket interface
- クライアントサーバモデル
- ライブラリ
- 開発環境

NEC

socket interface

- BSD 系UNIXでは，socket と呼ばれる一連のシステムコール群を用いて通信する
- 特徴
  - アドレス形式に依存しない
  - サーバ・クライアントモデル
  - プロトコル非依存
    - TCP/IP
    - XNS
    - ：

NEC

socket interface (2)

socket の生成	**socket**
名前付け	**bind**
接続受理準備	**listen**
接続要求	**connect**
接続要求受理	**accept**

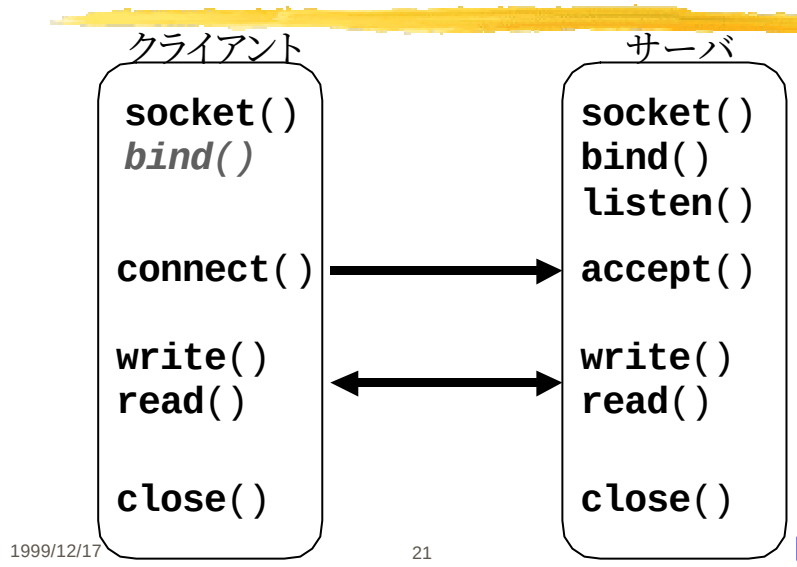
NEC

socket interface (3)

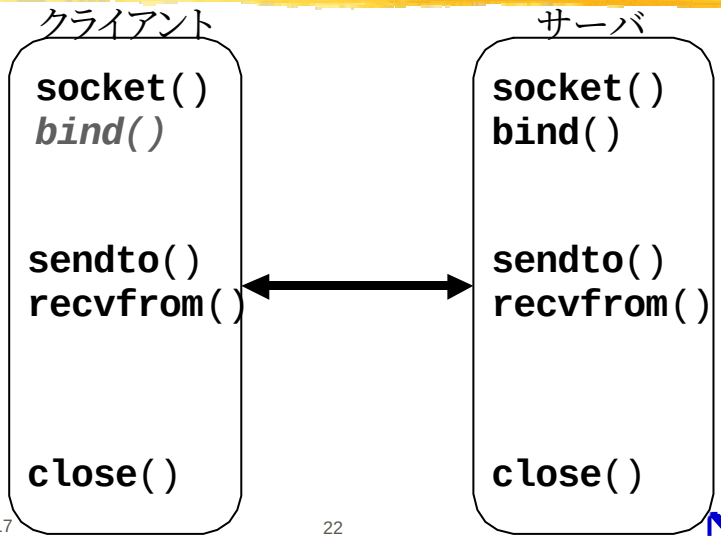
データ転送	**read, write** **recv, send** **recvfrom, sendto**
切断	**shutdown, close**
その他	**ioctl,socketpair,** **setsockopt,getsockopt,** **getsockname,getpeername**

NEC

socket を用いた通信(TCP)



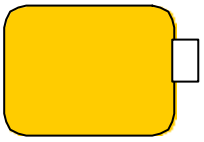
socket を用いた通信(UDP)

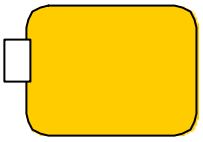


socket の生成

- 通信の口(socket)を作る

```
#include <sys/types.h>
#include <sys/socket.h>

int socket(proto_family, type, proto);
int proto_family, type, proto;
```





NEC

socket の生成 (2)

proto_family: プロトコルファミリ

PF_INET　　インターネットドメイン

他に PF_UNIX, PF_ISO, PF_INET6 などがある

type:　通信のタイプ

SOCK_STREAM　ストリーム(PF_INET の場合, TCP)

SOCK_DGRAM　データグラム(PF_INET の場合, UDP)

proto: プロトコル番号

0 にすると適当な番号に(システムが) 割り当ててくれる

sd:　　socket 記述子

NEC

socket の生成 (3)

- TCP の socket を生成

```
sd = socket(PF_INET, SOCK_STREAM, 0);
```

- UDP の socket を生成

```
sd = socket(PF_INET, SOCK_DGRAM, 0);
```


名前付け

- socket に名前をつける
  - 通信端点（IPアドレス,トランスポート,ポート番号)

```
#include <sys/types.h>
#include <sys/socket.h>

int bind(sd, name, namelen);
int sd, namelen;
struct sockaddr *name;
```

名前付け (2)

sd socket 記述子
name 名前へのポインタ
namelen 名前の長さ

name は sockaddr_in 構造体へのポインタとなる（文字列ではない）

sockaddr_in 構造体

- IPアドレスとポート番号
  - トランスポートは，socket 生成時に指定済

/usr/include/netinet/in.h

```
struct sockaddr_in {
    u_char  sin_len;  /* 構造体の長さ */
    u_char  sin_family; /* PF_INET */
    u_short sin_port; /* ポート番号 */
    struct  in_addr sin_addr; /*アドレス*/
    char    sin_zero[8]; /*詰物*/
};
struct in_addr {
    u_long  s_addr;         /* アドレス */
};
```

sockaddr_in 構造体 (2)

- アドレス，ポート番号はネットワークバイトオーダで指定する

```
struct sockaddr_in server;
server.sin_len = sizeof(server);
server.sin_family = PF_INET;
server.sin_port = htons(80);
server.sin_addr = htonl(0x0a030201);/*10.3.2.1*/
```

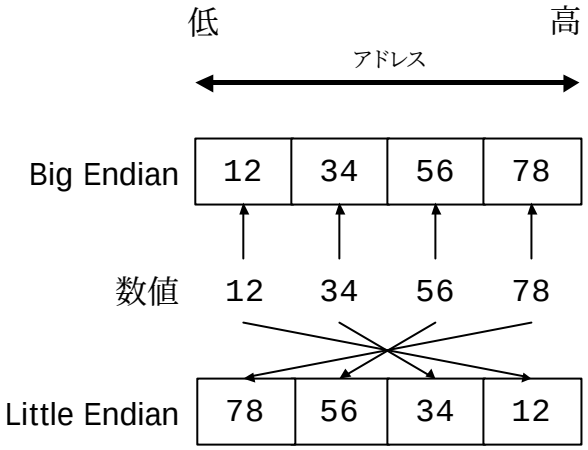

NEC

バイトオーダ

- ネットワークバイトオーダ
  - ネットワーク上を流れるデータのバイト順序
    - すべてのネットワークにおいて同一
  - Big Endian
- ホストバイトオーダ
  - CPU が扱うデータのバイト順序
  - Little Endian: VAX, 80x86, Pentium
  - Big Endian: 680x0, SPARC
  - Config で変わる: MIPS

NEC

バイトオーダ (2)



アドレス，ポート番号

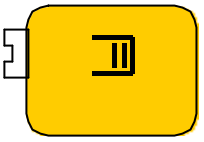
- アドレスに INADDR_ANY を指定すると，自ホストアドレスと解釈される
- ポート番号に 0 を指定するとシステムが適当なポート番号を割り当ててくれる
  - 通常，クライアントは **bind()**を実行しなくてよい

接続受理の準備

- サーバは接続要求受理の準備をおこなう

```
#include <sys/types.h>
#include <sys/socket.h>

int listen(sd, maxqueue);
int sd, maxqueue;
```

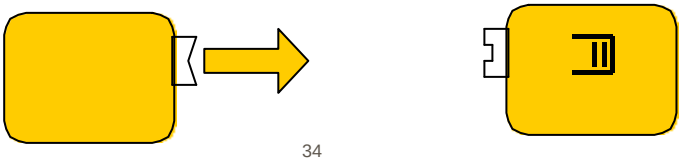
sd socket 記述子
maxqueue 受信受付キューの長さ



接続要求

- クライアントは，接続を要求する

```
#include <sys/types.h>
#include <sys/socket.h>

int connect(sd, name, namelen);
int sd, namelen;
struct sockaddr *name;
```



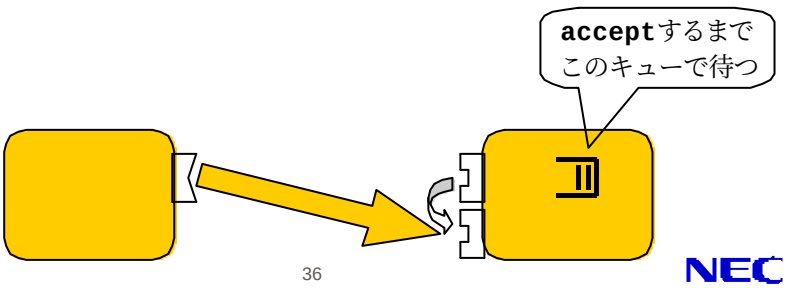
接続要求 (2)

sd socket 記述子
name 接続相手の名前へのポインタ
namelen 接続相手の名前の長さ

接続要求の受理

- サーバは，接続要求を受け付ける

```
#include <sys/types.h>
#include <sys/socket.h>

int accept(sd, name, namelen);
int      sd, *namelen;
struct sockaddr *name;
```



接続要求の受理 (2)

sd	socket 記述子
name	相手の名前を格納する領域へのポインタ
namelen	相手の名前の長さを格納する領域へのポインタ

- **accept** は新しい socket を作成する.
  - 以降の通信は,新しいsocket で行う

これで，接続が確立

NEC

データ転送

- UNIXの通常の入出力と同じ
  - **read, write** を使う
    - **recv, send, …**
  - クライアントは，socket 記述子
  - サーバは **accept()** の戻り値

NEC

データ転送-TCP

- データ受信用システムコール

```
#include <unistd.h>
ssize_t read(sd, buf, buflen);

#include <sys/types.h>
#include <sys/socket.h>
ssize_t recv(sd, buf, buflen, flags);
int sd, flags;
size_t   buflen;
void     *buf;
```


データ転送-TCP (2)

- データ送信用システムコール

```
#include <unistd.h>
ssize_t write(sd, buf, buflen);

#include <sys/types.h>
#include <sys/socket.h>
ssize_t send(sd, buf, buflen, flags);
int sd, flags;
size_t   buflen;
void     *buf;
```

データ転送-TCP (3)

sd	socket 記述子
buf	データバッファへのポインタ
buflen	データ長
flags	データ転送フラグ
MSG_OOB	帯域外データの処理
MSG_PEEK	バッファ中のデータを覗く
MSG_DONTROUTE	ルーティングをバイパスする
MSG_EOR	レコード境界を示す
MSG_EOF	データ転送の終了を示す
MSG_WAITALL	データが全部揃うまで待つ

NEC

データ転送-UDP

- データ転送用システムコール

```
#include <sys/types.h>
#include <sys/socket.h>

ssize_t sendto(sd, buf, buflen, flags, to,
          tolen);
ssize_t recvfrom(sd, buf, buflen, flags,
               from, fromlen);

int            sd, flags, tolen, *fromlen;
void           *buf;
ssize_t        buflen;
struct sockaddr *to, *from;
```


NEC

データ転送-UDP (2)

sd	socket記述子
buf	データバッファへのポインタ
to, from	相手の名前へのポインタ
flags	データ転送フラグ
buflen	データ転送長
tolen	名前の長さ
fromlen	名前の長さへのポインタ

NEC

切断

- 送受信の一方向，あるいは両方向のデータ転送の終了

```
#include <sys/type.h>
#include <sys/socket.h>

int       shutdown(sd, how);
int       sd, how;
```

sd	socket 記述子
how	切断方法
0	データの受信を終了する
1	データの送信を終了する
2	データの送受信を終了

NEC

切断 (2)

- 接続を切断し，ソケットを消滅

```
#include <unistd.h>

int close(sd);
int sd;
```

sd　　socket 記述子

- **shutdown** を利用せず **close** してよい

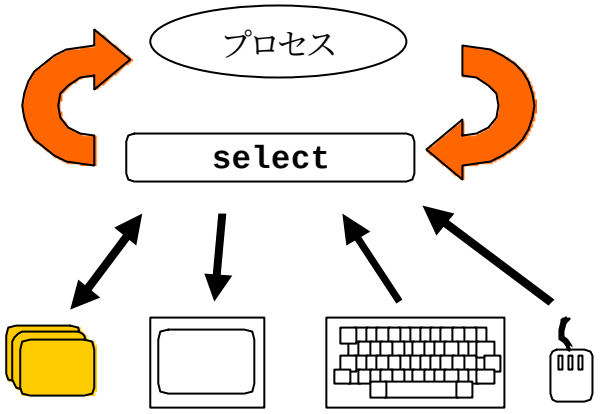
NEC

入出力の多重化

- **accept, read, recv** などのシステムコールは，データを受信するまでブロックしてしまう．
  複数の入出力を同時に見張りたいときには，
  select システムコールを使う

NEC

select



NEC

select システムコール

- 複数デバイスの入出力待ち

```
#include <sys/types.h>
#include <sys/time.h>
#include <sys/select.h>
int select(nfds, rfds, wfds, efds, tout);
int nfds;
fd_set *rfds, *wfds, *efds;
struct timeval *tout;
```


NEC

select システムコール (2)

nfds	調べるファイル記述子の数
rfds	入力用ファイル記述子の指定（ポインタ）
wfds	出力用ファイル記述子の指定（ポインタ）
efds	例外用ファイル記述子の指定（ポインタ）
tout	タイムアウトの指定(ポインタ) NULLの時は，無限に待つ
戻り値	入出力可能なファイル記述子数

select システムコール (3)

- **FD_SETSIZE** マクロ
  - システムがサポートする記述子の最大数
- ビット操作マクロ

  FD_SET(*fd, &fdset*); 記述子 ***fd*** を ***fdset*** にセットする

  FD_CLR(*fd, &fdset*); 記述子 ***fd*** を ***fdset*** からクリアする

  FD_ISSET(*fd, &fdset*); 記述子 ***fd*** が ***fdset*** に
  - セットされている場合 → 0 以外を返す
  - セットされていない場合 → 0 を返す

  FD_ZERO(*&fdset*); 記述子の集合を空にする

```
int fd;
fd_set fdset;
```


NEC

select システムコール (4)

- 使用例: 標準入力と socket sd の入力を待つ

```
fd_set rd;
for (;;) {
  FD_ZERO(&rd);
  FD_SET(fileno(stdin), &rd);
  FD_SET(sd, &rd);
  select(FD_SETSIZE, &rd, NULL, NULL, NULL);
  if (FD_ISSET(fileno(stdin), &rd)) {
   /* stdin の処理*/
  }
  if (FD_ISSET(sd, &rd)) {
   /* sd の処理*/
  }
}
```

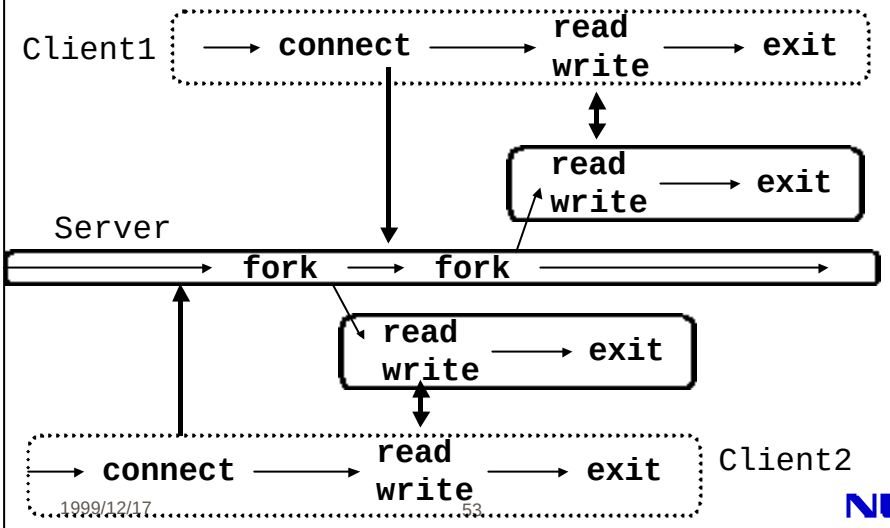

NEC

クライアント・サーバモデル

- ネットワークアプリケーションの基本的なモデル
- サーバ
  - 公示したサービスに対して要求を待ち，要求者に対してサービスを提供する（デーモンと呼ばれることも）
- クライアント
  - サーバに対してサービスを要求し，サーバからサービスを受ける

NEC

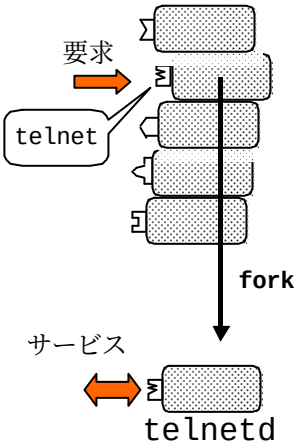
クライアント・サーバモデルの例



NEC

UNIX でのサーバ

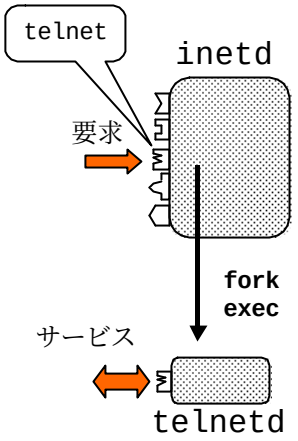
- サービスの公示
  - /etc/services 中にサービス名，ポート番号，プロトコル番号で示してある.
  - サーバは公示したポートを監視する
- サービスの要求
  - クライアントはサーバが存在するホストのインターネットアドレスとポート番号の組でサービスの要求を出す



NEC

inetd

- スーパサーバ
  - 提供するサービスがたくさんあると，プロセスが，たくさん必要
    - 資源の無駄
  - 一括してポートを監視
  - /etc/inetd.conf
- クライアントとの通信は標準入出力を使う
  - サーバプログラムが（少し）楽



ライブラリ

- getXbyY
  - ホスト名 ⇔ IPアドレス
  - ネットワーク名 ⇔ ネットワークアドレス
  - プロトコル名 ⇔ プロトコル番号
  - サービス名 ⇔ ポート番号
- インタネットアドレスの操作
- バイトオーダ変換

gethostbyname, gethostbyaddr

- ホスト名⇒IP アドレス

```
#include <netdb.h>
struct hostent *gethostbyname(name);
char *name;
```

- IPアドレス⇒ホスト名

```
#include <netdb.h>
struct hostent *gethostbyaddr(addr, len,
  type);
char *addr;
int len;
int type;
```

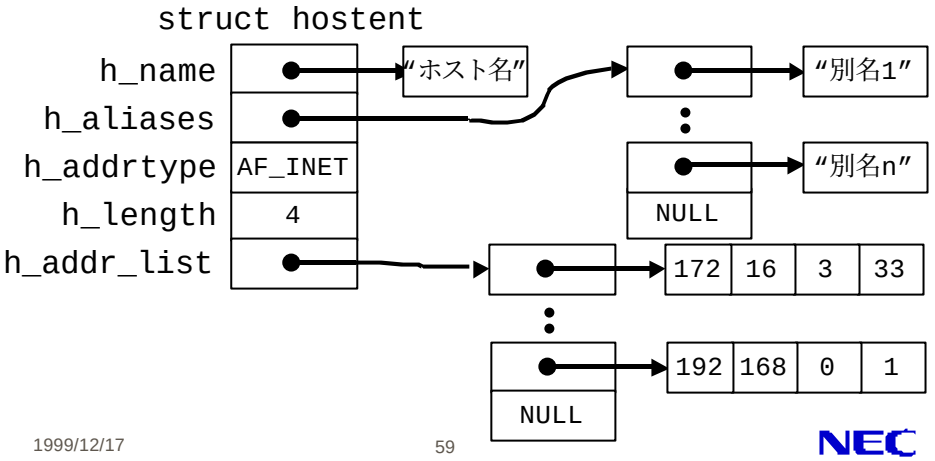

NEC

struct hostent

```
struct hostent {
  char    *h_name;     /*ホスト名*/
  char    **h_aliases;/*ホストの別名*/
  int     h_addrtype; /*アドレスタイプ(AF_INET)*/
  int     h_length;   /*アドレス長*/
  char    **h_addr_list; /*アドレスのリスト*/
};
```


NEC

struct hostent (2)



getnetbyname, getnetbyaddr

- ネットワーク名⇒ネットワークアドレス

```
#include <netdb.h>
struct netent *getnetbyname(name);
char *name;
```

- ネットワークアドレス⇒ネットワーク名

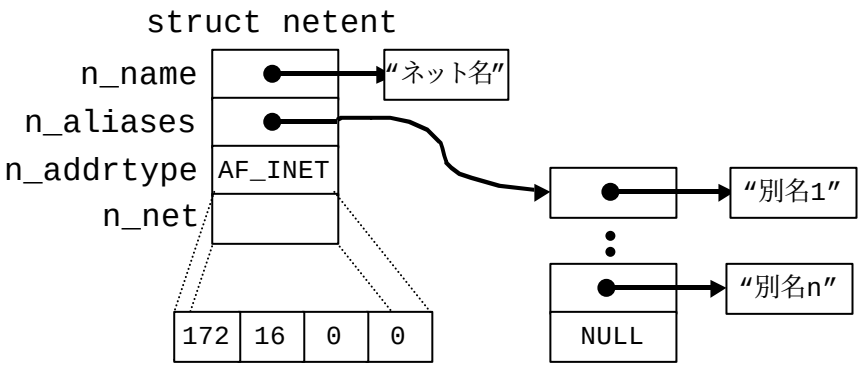
```
#include <netdb.h>
struct netent *getnetbyaddr(net, type);
long net;
int type;
```

struct netent

```
struct netent {
  char     *n_name;      /*ネットワーク名*/
  char     **n_aliases;  /*別名のリスト*/
  int      n_addrtype;   /*アドレスタイプ*/
  unsigned long n_net;   /*ネットワークアドレス*/
};
```


NEC

struct netent (2)



NEC

getprotobyname, getprotobynumber

- プロトコル名⇒プロトコル番号

```
#include <netdb.h>
struct protoent *getprotobyname(name);
char *name;
```

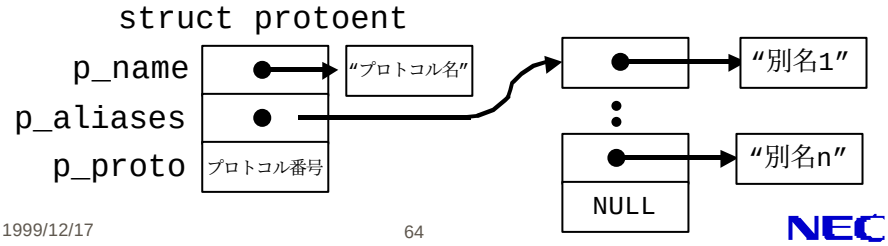
- プロトコル番号⇒プロトコル名

```
#include <netdb.h>
struct protoent *getprotobynumber(proto);
int proto;
```


NEC

struct protoent

```
struct protoent {
  char    *p_name;      /*プロトコル名*/
  char    **p_aliases;  /*別名*/
  int     p_proto;      /*プロトコル番号*/
};
```



NEC

getservbyname, getservbyport

- サービス名⇒ポート番号

```
#include <netdb.h>
struct servent *getservbyname(name, proto);
char *name;
char *proto;
```

- ポート番号⇒サービス名

```
#include <netdb.h>
struct servent *getservbyport(port, proto);
int port;
char *proto;
```

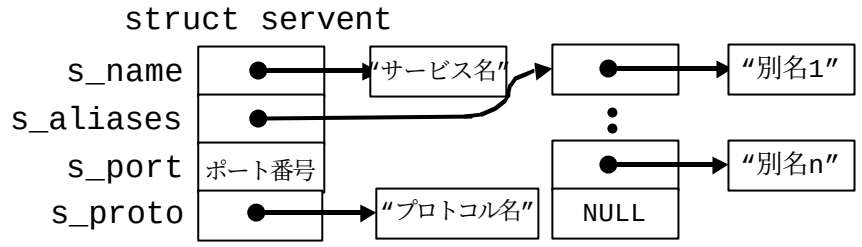

NEC

struct servent

```
struct servent {
  char      *s_name;     /*サービス名*/
  char      **s_aliases; /*別名のリスト*/
  int       s_port;      /*ポート番号*/
  char      *s_proto;    /*プロトコル名*/
};
```


NEC

struct servent (2)



インターネットアドレスの操作

- アドレス表現⇒IPアドレス

```
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int inet_aton(str, addr);

char *str; /* eg: “10.0.0.1” */
struct in_addr *addr;

in_addr_t inet_addr(str);

char *str; /* eg: “10.0.0.1” */
```

インターネットアドレスの操作 (2)

- IPアドレス⇒アドレス表現

```
#include <sys/types.h>
#include <netine/in.h>
#include <arpa/inet.h>

char *inet_ntoa(addr);
struct in_addr *addr;
```


インターネットアドレスの操作 (3)

- アドレス表現⇒ IPアドレス

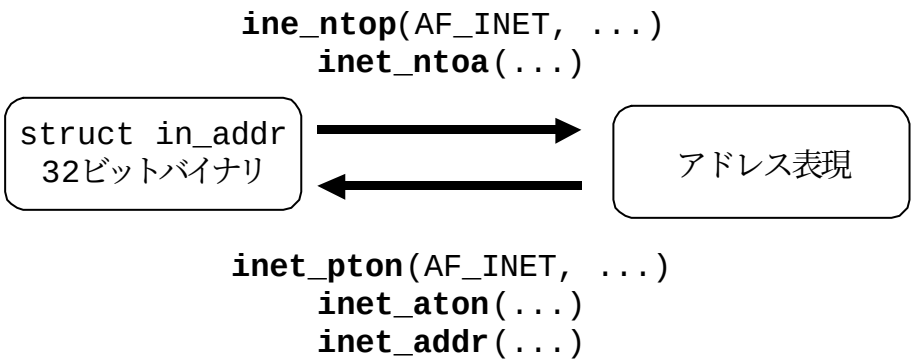
```
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
int inet_pton(family, str, addr)

int family;  /* AF_INET */
char *str;   /* eg: “10.0.0.1” */
void *addr;
```

インターネットアドレスの操作 (4)

- IPアドレス⇒アドレス表現

```
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
char *inet_ntop(family, addr, str, len);
int family;  /* AF_INET */
void *addr;
char *str;   /* string */
size_t len;  /* length of str */
```


インターネットアドレスの操作 (5)



バイトオーダの変換

- ホストバイトオーダ⇒ネットワークバイトオーダ

```
#include <sys/param.h>
u_long htonl(hostlong);
u_long hostlong;

u_short htons(hostshort);
u_short hostshort;
```


バイトオーダの変換 (2)

- ネットワークバイトオーダ⇒ホストバイトオーダ

```
#include <sys/param.h>
u_long ntohl(netlong);
u_long netlong;

u_short ntohs(netshort);
u_short netshort;
```

開発環境

- BSD 系では特にライブラリの指定は不必要
- System V 系では，ライブラリの指定が必要な場合も
  - `-lsocket`
- perl

NEC

Windows 編

- インターネット概要(石井)
- ネットワークプログラミング
  - UNIX編 (石井)
  - WINDOWS編(日比野)
  - JAVA編(日比野)
- 実例紹介(日比野･石井)
  - SMTP, POP
- まとめ

Windowsの開発環境

- C++(C)
  - Visual C++ (Microsoft)
  - C++ Builder (Imprise)等
- Delphi、Visual Basic等

Network API

- Winsock DLL
- WinInet
- Winsock Control
- Netscape、Internet Explorer

Windowsのライブラリ

- DLL
- COM/OCX
- DDE

DLL

- Dynamic Link library
  - 基本的に拡張子がDLL
  - Windows API(WIN32)もDLLで実装。
- オブジェクト(コード)の共有化
- バージョンの混乱
  - MFC42.DLL
  - W2Kで対応？

COM/OCX

- ActiveX Control
- Component Object Model
  - xxxx.dll/xxxx.ocx/xxxx.exe
- ソフトウェア部品

WinInet

- 以下のプロトコルをサポートするDLL
  - HTTP
  - FTP
  - Gopher

Winsock Control

- COM(Component Object Model)
- 多くの処理系で利用可能
  - Visual Basic
  - Visual Basic for Applications(VBA)
    - Excel,Word,Access...

Netscape,Internet Explorer

- IEはWEBブラウザコントロールとして使用可能。
- NetscapeもDDE経由で使用できる。

Windowsのプログラミングスタイル

- メッセージドリブン
  - ウインドウメッセージの処理
  - 定型的な処理はクラスライブラリが隠蔽
    - MFC等
- 多くのAPIがWindowを必要とする。
  - 通知にウインドウメッセージを使う。
    - winsock等

WinSock(1)

- Windows Sockets
- Windows 3.0から実装
  - 当初はベンダがネットワークカードやネットワークソフトウェアにバンドル。
  - Windows95以降はOSに標準バンドル。
- 最新はV2.0
  - wsock32.dll
  - Windows95はV1.1が標準

WinSock(2)

- Winsock API = Berkeley-style API+Windows-style API+その他
- Berkeley-style API
  - socket(),connect()...
- Windows-style API
  - Berkeley-style APIに対応するAPI
    - WSASocket(),WSAConnect()...
    - Windows-style APIは、今回のセミナの便宜上の名称。
- その他
  - 初期化API等

初期化API

- WSAStartup()
  - 初期化、バージョンチェック
- WSACleanup()
  - winsockの解放

Windows-style API (1)

- 非同期API
- WSAASync...
- ウインドウメッセージによる通知機構
  - 例：int WSAAsyncSelect(SOCKET s, HWND hWnd, UINT wMsg, long lEvent);
    - イベントマスク(lEvent)で指定したイベントが発生すると、ウインドウ(hWnd)にメッセージwMsgを送信する。

Windows-style API (2)

- 同期オブジェクトによる同期機構
  - WSAEventSelect
  - WSAWaitForMultipleEvents...
- Windowハンドルを必要としない。
- ただし、Winsock2以降。
  - Windows95では標準では使用できない。

Hungarian notation

- prefixに型をつける
  - hWnd(h=HANDLE)
  - lEvent(l=long)
  - p(pointer)...
- クラス(構造体)のメンバーにはm_を付加
  - m_hWnd,m_nSize

イベント

- FD_ACCEPT　接続確認通知
- FD_CLOSE　ソケットが閉じられたときの通知
- FD_CONNECT　接続結果通知
- FD_OOB　帯域外のデータ到達通知
- FD_WRITE　書き込み準備完了通知
- FD_READ　読み込み準備完了通知

WSAAsyncSelect

- selectの非同期版
- 通知がWindows Message

WSAAsyncGetXByY

- YからXを取得する。
  - WSAAsyncGetHostByAddr (gethostbyaddr)
  - WSAAsyncGetHostByName (gethostbyname)
  - WSAAsyncGetServByName (getservbyname)
  - WSAAsyncGetProtoByName (getprotobyname)
  - WSAAsyncGetProtoByNumber (getprotobynumber)
  - WSAAsyncGetServByName (getservbyname)
  - WSAAsyncGetServByPort (getservbyport)

WSAAsyncGetHostByName

```
HANDLE WSAAsyncGetHostByName (
    HWND hWnd, // window handle
    unsigned int wMsg,  // message
    const char * name,  // [in]host name
    char * buf,  // [out] HOSTENT
    int buflen //  length of buf
    );
```


WSACancelAsyncRequest

- WSAAsyncGetXByYのキャンセル。

エラー処理

- WSAGetLastError()
- WSASetLastError()
- エラーコードはunixとは異なる。
  - winsock.h
  - WSAENETDOWN 等“WSAE”が先頭につく。

例：CSocket

- MFC(Microsoft Foundation Classes)で実装されているSocketクラス
- CAsycnSocket/CSocket
- 非表示のウインドウを使用。
  - Windowsプログラミングの常道
    - WM_TIMER等

CAsyncSocket

- 非同期型
  - メソッド呼び出しはただちに完了する。イベントが発生するとコールバック関数が呼ばれる。
- コールバック関数
  - OnAccept/OnClose
  - OnConnect/OnOutOfBandData
  - OnRecieve()/OnSend()

CSocket

- CAsyncSocketの派生クラス
- 同期型
  - 処理が完了するまではメソッド呼び出しは戻ってこないので、プログラミングは容易。
- CArchiveとの連動

問題点(1)

- マルチスレッド
  - スレッド間の受け渡しはできない。
    - SOCKETハンドルを受け渡して、Attach/Detachを使用する。
    - CWndも同様。

```
SOCKET hSocket;
CSocket socket;
socket.Attach(hSocket);
...
hSocket = socket.Detach();
```


問題点(2)

- VC++6.0ではマルチスレッドで動作させると落ちる。
  - http://support.microsoft.com/support/kb/articles/q193/1/01.asp
    - Q193101 BUG: Unhandled Exception Using MFC Sockets in Visual C++ 6.0
    - 対策も書いてあります。
    - SP1(Service Pack 1)を待ちましょう。

SP1(Service Pack 1)を待ちましょう。

- ...と思ったら直っていなかった。
- SP5でも直っていなかった。

問題点(3)

- Connect()は同期API(gethostbyaddr()等)を使用している。
  - Connect()は使わないで非同期APIを使用する。

開発スタイル

- 4パターン
  - a.Berkeley-style API+シングルスレッド
  - b.Windows-style API+シングルスレッド
  - c.Berkeley-style API +マルチスレッド
  - d.Windows-style API +マルチスレッド
- 書籍等ではb,dを勧めることが多いが、私はcがお勧め。

Berkeley-style API+シングルスレッド

- ブロッキング中、他の操作ができない。
  - ウインドウの再描画等。
    - non blocking modeにして、イベントループを挿入する方法もある。
- Windowsプログラミングでは非現実的

Windows-style API+シングルスレッド

- ブロックしないので、「Berkeley-style API+シングルスレッド」の問題は回避できる。
- 複数のセッションを扱う場合などはプログラムが複雑になる傾向がある。
- unix等からの移植は面倒。
  - 全面書き直しに近い。

Berkeley-style API+マルチスレッド

- バックグラウンドで通信スレッドを動作させる。
- unix等からの移植は簡単。
  - ソースの共有化
  - OpenLDAP,PGP...
- マルチスレッド特有のプログラミングの面倒さがある。
- Windows3.1では使えない。
  - 忘れてしまうのが吉。

Windows-style API+マルチスレッド

- プログラムの構造はシンプルになる。
- CSocketを使うと結果的にそうなる。

Java編

- インターネット概要(石井)
- ネットワークプログラミング
  - UNIX編 (石井)
  - WINDOWS編(日比野)
  - JAVA編(日比野)
- 実例紹介(日比野・石井)
  - SMTP, POP
- まとめ

概要

- Javaは標準でSocketサポート
- 文字列の扱いに注意
- java.net.Socket/ java.net.ServerSocket...
- Servlet API
  - javax.servlet
  - http

プログラミングスタイル

- 同期型
  - 処理が完了するまでは呼び出しは戻ってこない。
- マルチスレッドが必須。
  - Javaはマルチスレッドを標準でサポート。

文字列(String)

- Unicode
- 多くの場合、Unicode<->byte[]の変換が必要。

```
String#getBytes(String enc)
String command = “RETR 1¥r¥n”;
out_stream.write(command.getBytes(“8859_1”));

byte[] bytes = new byte[255];
in_stream.readLine(bytes, 0, 255);
String recv = new String(bytes, “JIS”);
```


encoding

- 代表的なencoding
  - JIS : iso2022_jp
  - SJIS : Shift_jis
  - EUCJIS : EUC_JP
  - 8859_1 : iso8859_1
- 日本語の自動変換には“JisAutoDetect”
- byte列の展開には、“8859_1”を使用。

接続

```
int POP3_PORT = 110;
Socket socket = new Socket("hostname", POP3_PORT);
```


読み込み/書き込み(1)

```
import java.io.*;
... // 接続処理
BufferedInputStream in_stream = BufferedInputStream(socket.
    getInputStream());
BufferedOutputStream out_stream = BufferedOutputStream (socket.
    getOutputStream());
int ch = in_stream.read();
out_stream.write(ch);
```


Orangesoft Inc.

読み込み/書き込み(2)

```
import java.io.*;
... // 接続処理
BufferedInputReader reader = BufferedInputReader(new
    InputStreamReader(socket. getInputStream()));
BufferedOutputWriter writer = BufferedOutputWriter(new
    OutputStreamWriter(socket. getOutputStream()));
int ch = reader.read();
int ch = reader.write();
```


Reader(Writer)とStream

- メッセージ系の処理ではReaderは使いにくい。
- メッセージごとにcharsetは異なる。
  - MIMEマルチパートの場合、メッセージの中でcharsetは変わる。
  - 従って、メッセージを扱う場合はReaderを使った文字変換は使用できない。

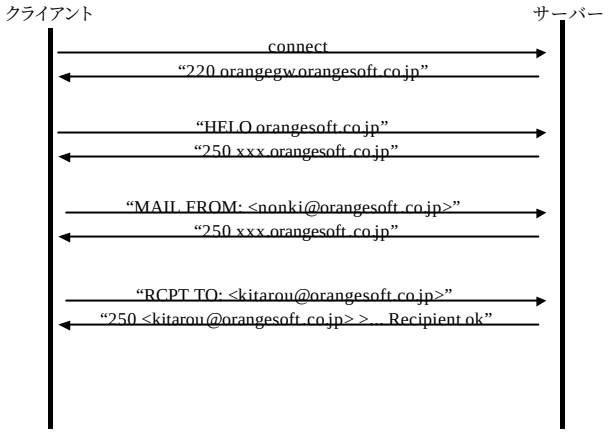
実例紹介

- インターネット概要(石井)
- ネットワークプログラミング
  - UNIX編 (石井)
  - WINDOWS編(日比野)
- 実例紹介(日比野･石井)
  - SMTP, POP
- まとめ

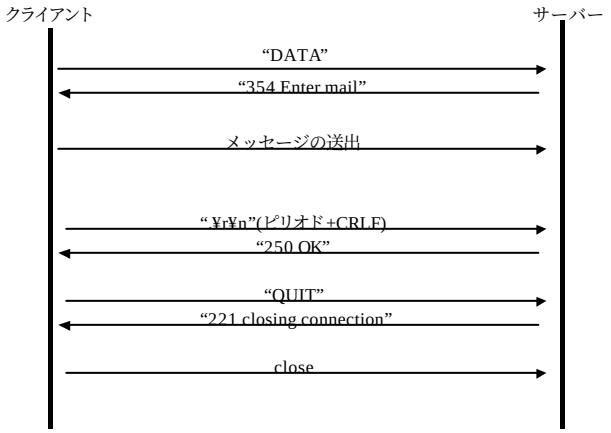
サンプル

- SMTP
  - SIMPLE MAIL TRANSFER PROTOCOL
  - RFC821
- POP3
  - POST OFFICE PROTOCOL
  - RFC1939

SMTP(1)


クライアント
サーバー
connect
“220 orangegw.orangesoft.co.jp”
“HELO orangesoft.co.jp”
“250 xxx.orangesoft.co.jp”
“MAIL FROM: <nonki@orangesoft.co.jp>”
“250 xxx.orangesoft.co.jp”
“RCPT TO: <kitarou@orangesoft.co.jp>”
“250 <kitarou@orangesoft.co.jp> >... Recipient ok”

SMTP(2)


クライアント
サーバー
“DATA”
“354 Enter mail”
メッセージの送出
“¥r¥n”(ピリオド+CRLF)
“250 OK”
“QUIT”
“221 closing connection”
close

SMTPコマンド

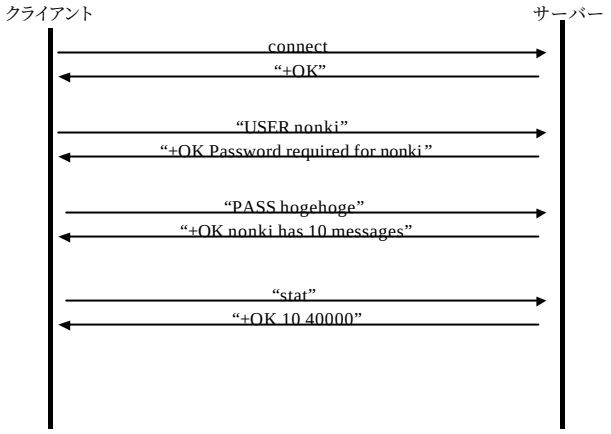
- コマンド一覧(抜粋)
  - HELO　使用開始
  - MAIL　メール送信の開始
  - RCPT　送り先メールアドレスの指定
  - DATA　メール本文の送信開始
  - RSET　リセット
  - NOOP　何もしない
  - QUIT　接続の終了

SMTPレスポンス

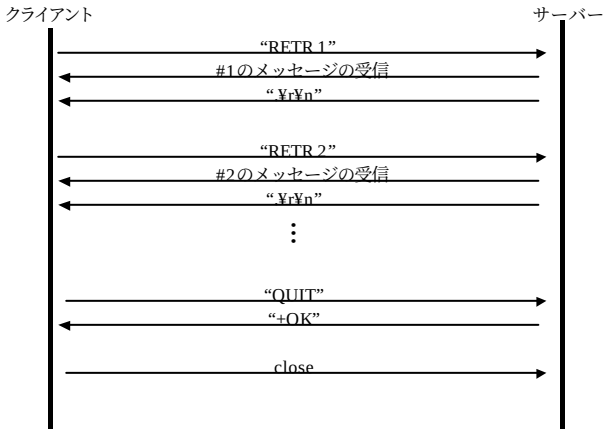
- 3桁(xyz)の数字で表現
  - 1yz 予備的な肯定
  - 2yz 肯定的な完了
  - 3yz 中間的な肯定
  - 4yz 一時的な否定
  - 5yz 永久的な否定
- 複数行の応答

```
123-aaaa
123-bbbb
123 ccccc
```

POP3(1)


クライアント
サーバー
connect
“+OK”
“USER nonki”
“+OK Password required for nonki”
“PASS hogehoge”
“+OK nonki has 10 messages”
“stat”
“+OK 10 40000”

POP3(2)


クライアント
サーバー
“RETR 1”
#1のメッセージの受信
“ ¥r¥n”
“RETR 2”
#2のメッセージの受信
“ ¥r¥n”
“QUIT”
“+OK”
close

POP3

- コマンド一覧
  - STAT　メールボックスのメール数とサイズの所得
  - LIST　ここのメールのサイズの所得
  - RETR　指定したメールの取出し
  - DELE　指定したメールの削除
  - NOOP　何もしない
  - RSET　操作の取消し
  - QUIT　接続の終了
  - TOP　指定したメッセージのヘッダと本文を指定した行数所得する
  - UIDL　ここのメールのサーバ内でのIDを所得する
  - USER　ユーザ認証時のユーザ名の送信
  - PASS　ユーザ認証時のパスワードの送信
  - APOP　MD5で暗号化されたユーザ認証
- レスポンス
  - +OK 成功　-ERR 失敗

特徴

- テキストベース
  - 行単位の処理
    - 転送データ量は予測できない。
    - CRLF(0x0d,0x0a)がセパレータ

まとめ

- インターネット概要(石井)
- ネットワークプログラミング
  - UNIX編 (石井)
  - WINDOWS編(日比野)
  - JAVA編(日比野)
- 実例紹介(日比野･石井)
  - SMTP, POP
- まとめ

まとめ

- インターネット
  - 異なったメディアを統合した論理的なネットワーク
  - IPアドレス、トランスポート、ポート番号
- UNIX
  - socket interface
  - 入出力の多重化:select
  - サーバ：fork,inetd

まとめ(2)

- Windows
  - Windows-style APIとunix互換のBerkeley-style APIの２種類をもつ。
  - Windows-style APIはメッセージベースの非同期API
- Java
  - Socketを標準でサポート
  - 同期型のAPI

Resources

- UNIX Network Programming Volume 1 Second Edition, Networking APIs: Sockets and XTI (W.Richard Stevens 著)
  - Prentice Hall (ISBN 0-13-490012-X)
- UNIX ネットワークプログラミング 第2版 Vol.1 ネットワークAPI：ソケットとXTI（篠田陽一 訳）
  - トッパン (ISBN 4-8101-8612-1)

NEC

Resources (2)

- インターネットを256倍使うための本 Vol. 2, (志村 拓, 榊 隆, 岩井 潔 共著)
  - アスキー出版局 (ISBN 4-7561-1927-1)
- マニュアルページ(**man** コマンド)

Resources (3)

- WinSock2.0プログラミング
  - ソフトバンク(ISBN4-7973-0688-2)
- インターネットRFC事典
  - アスキー(ISBN4-7561-1888-7)

Resources (4)

- Java House Mailing List
  - http://java-house.etl.go.jp/ml/
- Javaカンファレンス
  - http://www.java-fj.or.jp/

Resources (5)

- MSDN(Microsoft Developer Network)
  - http://www.microsoft.com/japan/developer/
- MSJ(J) (Microsoft Systems Journal)
- NiftyServe FWINDCフォーラム

```c
/*
 * daytimed サンプルコード
 *
 * Copyright 1999, Shuji Ishii, shuji@ccs.mt.nec.co.jp
 */

#include <stdio.h>
#include <string.h>
#include <unistd.h>

#include <sys/types.h>
#include <sys/time.h>
#include <sys/wait.h>
#include <signal.h>

#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <time.h>

/* functions */
static void print_daytime(int sd);
static void sigchld(int sig);

/*
 * main
 */
void main(void)
{
    int sd, snew;
    struct sockaddr_in sin;

    /*
     * シグナルハンドラの登録
     * 子プロセスが終了すると, SIGCHLDを受信する
     */
    (void)signal(SIGCHLD, sigchld);

    if ((sd = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket");
        exit(1);
    }

    sin.sin_len = sizeof(sin);
    sin.sin_family = AF_INET;
    sin.sin_port = htons(8888);
    sin.sin_addr.s_addr = INADDR_ANY;

    if (bind(sd, (struct sockaddr *)&sin, sizeof(sin)) < 0) {
        perror("bind");
        exit(2);
    }

    if (listen(sd, 5) < 0) {
        perror("listen");
        exit(3);
    }

    for (;;) {
        /* 要求待ち */
        if ((snew = accept(sd, NULL, NULL)) < 0) {
            perror("accept");
            exit(4);
        }

        /* 子プロセスの起動 */
        switch (fork()) {
        case -1:    /* エラー */
            perror("fork");
            exit(5);

        case 0:     /* 子プロセス */
            close(sd); /* 要求待ちソケットを閉じる */
            print_daytime(snew);
            exit(0); /* 処理終了 */

        default:    /* 親プロセス */
            close(snew); /* 子プロセス用ソケットを閉じる */
        }
    }
}

/*
 * シグナルハンドラ
 * 子プロセスが終了すると呼び出される
 */
static void sigchld(int sig)
{
    (void)wait(NULL);
}

/*
 * 日付, 時刻を表示
 */
static void print_daytime(int sd)
{
    char strbuf[64];
    struct timeval tv;

    (void)gettimeofday(&tv, NULL); /* 現時刻を取得 */

    /* 例: "Sun Oct 10 10:58:33 1999"*/
    (void)strftime(strbuf, sizeof(strbuf), "%a %b %d %T %Y\r\n",
                localtime(&tv.tv_sec)); /* 文字列に変換 */

    (void)write(sd, strbuf, strlen(strbuf));
}

/* ----- end of daytimed.c ----- */
```

```c
/*
 * メールを送信する(SMTP)/socket 版
 *
 * Copyright 1998, 1999, Shuji Ishii, shuji@ccs.mt.nec.co.jp
 */

#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define SMTP_RESULT_LEN (256)

typedef struct _smtpserver {
    FILE *in, *out;     /* サーバ入出力用 */
    char result[SMTP_RESULT_LEN]; /* 結果格納領域 */
} SMTPSERVER;

typedef struct _mail {
    char **headers;     /* ヘッダ */
    FILE *body;     /* 本文 */
} MAIL;

#define SMTP_OK (0)
#define SMTP_ERR (-1)

static int transaction(SMTPSERVER *smtp, char *command, MAIL *mail);

/*
 * SMTP サーバに接続する
 * 接続に成功すれば SMTPSERVER 構造体へのポインタを返す
 * エラーの場合 NULL を返す
 */
static SMTPSERVER *open_smtpserver(char *smtpserver)
{
    struct hostent *hp; /* ホストエントリ */
    struct protoent *pp;    /* プロトコルエントリ */
    struct servent *sp; /* サービスエントリ */
    int s = -1;     /* ソケット */
    struct sockaddr_in sin; /* サーバ */
    SMTPSERVER *smtp = NULL;
    char buf[128];

    /* smtp サーバのエントリを取得する */
    if ((hp = gethostbyname(smtpserver)) == NULL) {
        goto err;
    }
    /* tcp のエントリを取得する */
    if ((pp = getprotobyname("tcp")) == NULL) {
        goto err;
    }
    /* smtp サービスのエントリを取得する */
    if ((sp = getservbyname("smtp", "tcp")) == NULL) {
        goto err;
    }

    /* SMTPSERVER 構造体の確保 */
    if ((smtp = (SMTPSERVER *)malloc(sizeof(SMTPSERVER))) == NULL) {
        goto err;
    }

    /* socket を作る */
    if ((s = socket(PF_INET, SOCK_STREAM, pp->p_proto)) < 0) {
        goto err;
    }

    /* sockaddr_in 構造体を設定する */
    memset(&sin, 0, sizeof(sin));
    sin.sin_len = sizeof(sin);
    sin.sin_family = PF_INET;
    sin.sin_port = sp->s_port;
    memcpy(&sin.sin_addr, *(hp->h_addr_list), sizeof(sin.sin_addr));

    /* smtp サーバに接続する */
    if (connect(s, (struct sockaddr *)&sin, sizeof(sin)) < 0) {
        goto err;
    }

    /* smtp サーバへの通信路を fdopen で open する */
    smtp->in = fdopen(s, "r");
    smtp->out = fdopen(s, "w");

    /* greeting message を読み込む */
    if (transaction(smtp, NULL, NULL) != SMTP_OK) {
        fprintf(stderr, "open_smtpserver: %s\n", smtp->result);
        goto err;
    }

    return smtp;

    err:
    if (smtp) {
        free(smtp);
    }
    if (s >= 0) {
        close(s);
    }
    return NULL;
}

/*
 * smtp サーバへの接続を閉じる
 */
static void close_smtpserver(SMTPSERVER *smtp)
{
    fclose(smtp->in);
    fclose(smtp->out);
    free(smtp);
}

/*
 * コマンドを送信し、結果が OK なら SMTP_OK を、それ以外なら SMTP_ERR を
 * 返す
 *
 * 2xx: OK
 * 5xx: ERR
 * 3xx: need data
 */
static int transaction(SMTPSERVER *smtp, char *command, MAIL *mail)
{
    char buf[256];
```

```
125     char *newline;
126     int err = SMTP_OK;
127 
128     if (command != NULL) {
129         fprintf(smtp->out, "%s\r\n", command);
130         fflush(smtp->out);
131     }
132  getstat:
133     smtp->result[0] = '\0';
134     do {
135         fgets(buf, sizeof(buf), smtp->in);
136         if (smtp->result[0] == '\0') {
137             strncpy(smtp->result, buf, SMTP_RESULT_LEN);
138         }
139     } while (buf[3] == '-');
140 
141     switch (buf[0]) {
142     case '1':
143     case '2':
144     case '4':
145         break;
146 
147     case '3':       /* feed data */
148         /* ヘッダ */
149         while (*mail->headers != NULL) {
150             fprintf(smtp->out, "%s\r\n", *mail->headers);
151             mail->headers++;
152         }
153         fputs("\r\n", smtp->out);
154 
155         /* ボディ */
156         while (fgets(buf, sizeof(buf), mail->body) != NULL) {
157             if (buf[0] == '.') {
158                 /* 先頭が '.' ならエスケープ  */
159                 fputc('.', smtp->out);
160             }
161             if ((newline = strchr(buf, '\n')) != NULL) {
162                 *newline = '\0';
163                 fprintf(smtp->out, "%s\r\n", buf);
164             } else {
165                 fprintf(smtp->out, "%s", buf);
166             }
167         }
168         fputs(".\r\n", smtp->out);
169         fflush(smtp->out);
170 
171         goto getstat;
172         /*NOTREACHED*/
173 
174     case '5':
175         err = SMTP_ERR;
176         break;
177     }
178 
179     return err;
180 }
181 
182 void main(int argc, char **argv)
183 {
184     MAIL mail;
185     SMTPSERVER *smtp;
186     static char *headers[] = {
187         "From: <shuji>",
188         "To: <nonki>",
189         "Subject: test",
190         "Mime-Version: 1.0",
191         "Content-type: text/plain; charset=us-ascii",
192         "Content-Transfer-Encoding: 7bit",
193         NULL,
194     };
195 
196     mail.headers = headers;
197     mail.body = stdin;
198 
199     if ((smtp = open_smtpserver("mailserver")) == NULL) {
200         fprintf(stderr, "open_smtpserver: localhost\n");
201         exit(1);
202     }
203 
204     if (transaction(smtp, "HELO localhost", NULL) == SMTP_ERR) {
205         fprintf(stderr, "transaction: HELO; %s\n", smtp->result);
206         goto quit;
207     }
208     if (transaction(smtp, "MAIL FROM: <shuji>", NULL) == SMTP_ERR) {
209         fprintf(stderr, "transaction: MAIL FROM; %s\n", smtp->result);
210         goto quit;
211     }
212     if (transaction(smtp, "RCPT TO: <nonki>", NULL) == SMTP_ERR) {
213         fprintf(stderr, "transaction: RCPT TO; %s\n", smtp->result);
214         goto quit;
215     }
216     if (transaction(smtp, "DATA", &mail) == SMTP_ERR) {
217         fprintf(stderr, "transaction: DATA; %s\n", smtp->result);
218         goto quit;
219     }
220     if (transaction(smtp, "QUIT", NULL) == SMTP_ERR) {
221         fprintf(stderr, "transaction: QUIT; %s\n", smtp->result);
222         goto quit;
223     }
224 
225  quit:
226     close_smtpserver(smtp);
227     exit(0);
228 }
229 
230 /* ----- end of delivermail-socket.c ----- */
```

```c
/*
 * メールを受信する(POP)/socket 版
 *
 * Copyright 1998, 1999, Shuji Ishii, shuji@ccs.mt.nec.co.jp
 */

#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <pwd.h>

#define POP_RESULT_LEN (256)

typedef struct _popserver {
    FILE *in, *out;     /* POPサーバ入出力用 */
    char result[POP_RESULT_LEN]; /* 結果格納領域 */
} POPSERVER;

#define POP_OK (0)
#define POP_ERR (-1)

static int transaction(POPSERVER *pop, char *command);

/*
 * pop サーバに接続する.
 * 接続に成功すれば POPSERVER 構造体へのポインタを返す
 * エラーの場合 NULL を返す
 */
static POPSERVER *open_popserver(char *popserver)
{
    struct hostent *hp; /* ホストエントリ */
    struct protoent *pp;    /* プロトコルエントリ */
    struct servent *sp; /* サービスエントリ */
    int s = -1;     /* ソケット */
    struct sockaddr_in sin; /* サーバ */
    POPSERVER *pop = NULL;

    /* pop サーバのエントリを取得する */
    if ((hp = gethostbyname(popserver)) == NULL) {
        goto err;
    }
    /* tcp のエントリを取得する */
    if ((pp = getprotobyname("tcp")) == NULL) {
        goto err;
    }
    /* pop3 サービスのエントリを取得する */
    if ((sp = getservbyname("pop3", "tcp")) == NULL) {
        goto err;
    }

    if ((pop = (POPSERVER *)malloc(sizeof(POPSERVER))) == NULL) {
        goto err;
    }

    /* socket を作る */
    if ((s = socket(PF_INET, SOCK_STREAM, pp->p_proto)) < 0) {
        goto err;
    }

    /* sockaddr_in 構造体を設定する */
    memset(&sin, 0, sizeof(sin));
    sin.sin_len = sizeof(sin);
    sin.sin_family = PF_INET;
    sin.sin_port = sp->s_port;
    memcpy(&sin.sin_addr, *(hp->h_addr_list), sizeof(sin.sin_addr));

    /* pop サーバに接続する */
    if (connect(s, (struct sockaddr *)&sin, sizeof(sin)) < 0) {
        goto err;
    }

    /* pop サーバへの通信路を fdopen で open する */
    pop->in = fdopen(s, "r");
    pop->out = fdopen(s, "w");

    /* greeting message を読み込む */
    transaction(pop, NULL);

    return pop;

    err:
    if (pop) {
        free(pop);
    }
    if (s >= 0) {
        close(s);
    }
    return NULL;
}

/*
 * pop サーバへの接続を閉じる
 */
static void close_popserver(POPSERVER *pop)
{
    fclose(pop->in);
    fclose(pop->out);
    free(pop);
}

/*
 * コマンドを送信し、結果が +OK なら POP_OK を、それ以外なら POP_ERR を
 * 返す
 */
static int transaction(POPSERVER *pop, char *command)
{
    if (command != NULL) {
        fprintf(pop->out, "%s\r\n", command);
        fflush(pop->out);
    }
    fgets(pop->result, POP_RESULT_LEN, pop->in);

    return (pop->result[0] == '+') ? POP_OK : POP_ERR;
}

/*
 * pop サーバに login する
 */
static int pop_login(POPSERVER *pop, char *user, char *pass)
{
```

```
125     char buf[256];
126
127                 /* USER コマンド */
128     snprintf(buf, sizeof(buf), "USER %s", user);
129     if (transaction(pop, buf) == POP_ERR) {
130         return POP_ERR;
131     }
132
133                 /* PASS コマンド */
134     snprintf(buf, sizeof(buf), "PASS %s", pass);
135     if (transaction(pop, buf) == POP_ERR) {
136         return POP_ERR;
137     }
138
139     /*パスワードを消す*/
140     memset(buf, 0, sizeof(buf));
141
142     return POP_OK;
143 }
144
145 /*
146  * 現在のメッセージ数を返す
147  */
148 static int pop_msgcnt(POPSERVER *pop)
149 {
150     int msgcnt, ttlsize;
151
152                 /* STAT コマンド */
153     if (transaction(pop, "STAT") == POP_ERR) {
154         return -1;
155     }
156                 /* 結果を解析 */
157     if (sscanf(pop->result, "+OK %d %d", &msgcnt, &ttlsize) != 2) {
158         return -1;
159     }
160
161     return msgcnt;
162 }
163
164 /*
165  * 指定されたメッセージ番号を持つメールを out に指定された出力に
166  * 出力する
167  */
168 static int pop_retr(POPSERVER *pop, int msg, FILE *out)
169 {
170     char buf[256];
171
172                 /* RETR コマンド */
173     snprintf(buf, sizeof(buf), "RETR %d", msg);
174     if (transaction(pop, buf) == POP_ERR) {
175         return POP_ERR;
176     }
177
178     for (;;) {
179         fgets(buf, sizeof(buf), pop->in);
180
181         /* 行頭が '.'の時の処理 */
182         if (buf[0] == '.') {
183             if (buf[1] == '.') {
184                 fprintf(out, "%s", buf + 1);
185                 continue;
186             } else if (buf[1] == '\r' && buf[2] == '\n') {
187                 /* end of file */
188                 break;
189             }
190         }
191         fprintf(out, "%s", buf);
192     }
193
194     return POP_OK;
195 }
196
197 void main(int argc, char **argv)
198 {
199     POPSERVER *pop;
200     char *pass;
201     int msgcnt;
202     int i;
203
204     if ((pop = open_popserver("mailserver")) == NULL) {
205         fprintf(stderr, "error: %s\n", *argv);
206         exit(2);
207     }
208
209     if (pop_login(pop, "shuji", "password")) == POP_ERR) {
210         fprintf(stderr, "error: pop_login\n");
211         exit(3);
212     }
213
214     if ((msgcnt = pop_msgcnt(pop)) < 0) {
215         fprintf(stderr, "error: pop_msgcnt\n");
216         exit(4);
217     }
218
219     for (i = 1; i <= msgcnt; i++) {
220         if (pop_retr(pop, i, stdout) == POP_ERR) {
221             fprintf(stderr, "error: pop_retr\n");
222             break;
223         }
224     }
225
226     close_popserver(pop);
227     exit(0);
228 }
229
230 /* ----- end of fetchmail-socket.c ----- */
```

```
#!/usr/local/bin/perl5

#
# メールを送信する(SMTP)/perl5 Mail::Internet 版
#
# Copyright 1998,1999, Shuji Ishii, shuji@ccs.mt.nec.co.jp
#

##
## libnet-x.xx
## DataDumper-x.xx
## MailTools-x.xx
##

use Mail::Internet;
use Mail::Header;

@headers = (
        "From: <shuji>",
        "To: <nonki>",
        "Subject: test",
        "Mime-Version: 1.0",
        "Content-type: text/plain; charset=us-ascii",
        "Content-Transfer-Encoding: 7bit",
        );

$header = new Mail::Header(\@headers);
$msg = new Mail::Internet([<>], Header => $header);
$msg->smtpsend({Host => "mailserver"});

## end of delivermail-perl5-mailinternet.pl ##
```

```
#!/usr/local/bin/perl5

#
# メールを送信する(SMTP)/perl5 IO::Socket 版
#
# Copyright 1998,1999, Shuji Ishii, shuji@ccs.mt.nec.co.jp
#

use IO::Socket;

#
# SMTP サーバに接続する.
#
sub open_smtpserver
{
    local($smtpserver) = @_;

    $SMTP = IO::Socket::INET->new(PeerAddr => $smtpserver,
                  PeerPort => 'smtp(25)',
                  Proto    => 'tcp');
    unless ($SMTP) {
    die $!;
    }
    autoflush $SMTP 1;

    return &transaction(undef);
}

sub close_smtpserver
{
    close($SMTP);
}

#
# コマンドを送信し、結果を返す
#
# 2xx: OK
# 5xx: ERR
# 3xx: need data
#
sub transaction
{
    local($command, @headers) = @_;
    local($result, $h, $dot);

    if (defined($command)) {
    print $SMTP "$command\r\n";
    }

getstat:
    for (;;) {
    undef($result);

    do {
        $_ = <$SMTP>;
        chop; chop;     # drop '\r\n'
        unless ($result) {
        $result = $_;
        }
    } while (/^\d\d\d\-/);
    if (/^[124]/) {
        last;       # break;
    } elsif (/^3/) {    # データ
        # ヘッダ
        for $h (@headers) {
        print $SMTP "$h\r\n";
        }
        print $SMTP "\r\n";
        # 本文
        while (<>) {
        chop;       # drop '\n' (when unix)
        if (/^\./) {
            $dot = ".";
        } else {
            $dot = '';
        }
        print $SMTP "$dot$_\r\n";
        }
        print $SMTP ".\r\n";    # 本文の終り
        next getstat;
    } elsif (/^5/) {
        last;           # break
    }
    }

    return $result;
}

## main

@headers = (
        "From: <shuji>",
        "To: <nonki>",
        "Subject: test",
        "Mime-Version: 1.0",
        "Content-type: text/plain; charset=us-ascii",
        "Content-Transfer-Encoding: 7bit",
        );

if (&open_smtpserver("mailserver") !~ /^2/) {
    die $!;
}
$myname = chop `hostname`;
if (&transaction("HELO $myname") =~ /^5/) {
    die $!;
}
if (&transaction("MAIL FROM: <shuji>") =~ /^5/) {
    die $!;
}
if (&transaction("RCPT TO: <nonki>") =~ /^5/) {
    die $!;
}
if (&transaction("DATA", @headers) =~ /^5/) {
    die $!;
}
if (&transaction("QUIT") =~ /^5/) {
    die $!;
}

&close_smtpserver;
exit(0);

## end of delivermail-perl5-sock.pl ##
```

```
#!/usr/local/bin/perl5

#
# メールを受信する(POP)/perl5 Mail::POP3Client 版
#
# Copyright 1998, 1999, Shuji Ishii, shuji@ccs.mt.nec.co.jp
#

##
## POP3Client-x.xx
##

use Mail::POP3Client

$pop = new Mail::POP3Client("shuji", "password", "mailserver");
for ($i = 1; $i <= $pop->Count; $i++) {
    $_ = $pop->Retrieve($i);
    tr/\r//d;
    print "$_\n";
}

## end of fetchmail-perl5-mailpop3client.pl ##
----Next_Part(Sat_Nov__6_14:10:56_1999)----
```

```
#!/usr/local/bin/perl5

use IO::Socket;

#
# メールを受信する(POP)/perl5 IO::Socket 版
#
# Copyright 1998, 1999, Shuji Ishii, shuji@ccs.mt.nec.co.jp
#

#
# pop サーバに接続する.
#
sub open_popserver
{
    local($popserver) = @_;

    $POP = IO::Socket::INET->new(PeerAddr => $popserver,
                 PeerPort => 'pop3(110)',
                 Proto    => 'tcp');
    unless ($POP) {
    die $!;
    }

    autoflush $POP 1;

    return &transaction(undef);
}

sub close_popserver
{
    close($POP);
}

#
# コマンドを送信し、結果を返す
#
sub transaction
{
    local($command) = @_;
    local($result);

    if (defined($command)) {
    print $POP "$command\r\n";
    }

    $result = <$POP>;
    chop $result; chop $result;         # drop '\r\n';

    return $result;
}

sub pop_login
{
    local($user, $pass) = @_;
    local($result);

    if (($result = &transaction("USER $user")) =~ /^\-ERR/) {
    return $result;
    }
    return &transaction("PASS $pass");
}

sub pop_msgcnt
{
    local($msgcnt, $ttlsize);

    if (&transaction("STAT") =~ /^\+OK (\d+) (\d+)/) {
    return $1;
    }

    return -1;
}

sub pop_retr
{
    local($msg) = @_;
    local($result);

    if (($result = &transaction("RETR $msg")) =~ /^\-ERR/) {
    return $undef;
    }

    while (<$POP>) {
    chop; chop;     # drop '\r\n';
    if (/^\.(\..*)$/) {
        print "$1\n";
    } elsif (/^\.$/) {
        # eof
        last;
    } else {
        print "$_\n";
    }
    }

    return $result;
}

##
## main
##

&open_popserver("mailserver");
if (&pop_login("shuji", "password") =~ /^\-/) {
    die "login error\n";
}
$msgcnt = &pop_msgcnt;
if ($msgcnt < 0) {
    die "pop_msgcnt: $msgcnt\n";
}

for ($i = 1; $i <= $msgcnt; $i++) {
    if (!&pop_retr($i)) {
    die "pop_retr\n";
    }
}

&close_popserver;
exit(0);

## end of fetchmail-perl5-sock.pl ##
```

```cpp
//
//  Copyright (c) 1999 Orangesoft Inc.
//

#include "stdafx.h"
#include <winsock.h>
#include "resource.h"

#include "sendmail.h"
#include "recvmail.h"

#define MAX_LOADSTRING 100

// グローバル変数:
HINSTANCE hInst;                        // 現在のインスタンス
TCHAR szTitle[MAX_LOADSTRING];          // タイトル バー テキスト
TCHAR szWindowClass[MAX_LOADSTRING];    // タイトル バー テキスト

// このコード モジュールに含まれる関数の前宣言:
ATOM MyRegisterClass( HINSTANCE hInstance );
BOOL InitInstance( HINSTANCE, int );
LRESULT CALLBACK WndProc( HWND, UINT, WPARAM, LPARAM );
bool InitWinsock(WSADATA* pWsaData);

int APIENTRY WinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR     lpCmdLine,
                     int       nCmdShow )
{
    MSG msg;
    HACCEL hAccelTable;
    WSADATA wsadata;

    LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
    LoadString(hInstance, IDC_IW98_1, szWindowClass, MAX_LOADSTRING);
    MyRegisterClass( hInstance );

    if( !InitInstance( hInstance, nCmdShow ) )
    {
        return FALSE;
    }

    // Winsockの初期化
    if(!InitWinsock(&wsadata)){
        return FALSE;
    }

    hAccelTable = LoadAccelerators(hInstance, (LPCTSTR)IDC_IW98_1);

    while( GetMessage(&msg, NULL, 0, 0) )
    {
        if( !TranslateAccelerator (msg.hwnd, hAccelTable, &msg) )
        {
            TranslateMessage( &msg );
            DispatchMessage( &msg );
        }
    }

    // Winsockの解放
    WSACleanup();

    return msg.wParam;
}

// Winsockの初期化
bool InitWinsock(WSADATA* pWsaData)
{
    int nRequiredVersion = MAKEWORD(1,1);       // Version 1.1
    if(WSAStartup(nRequiredVersion, pWsaData) != 0){
        return false;
    }
    if(pWsaData->wVersion != nRequiredVersion){
        WSACleanup();
        return false;
    }
    return true;
}

//
//  関数: MyRegisterClass()
//
//  用途: ウィンドウ クラスの登録
//
//
ATOM MyRegisterClass( HINSTANCE hInstance )
{
    WNDCLASSEX wcex;

    wcex.cbSize = sizeof(WNDCLASSEX);

    wcex.style          = CS_HREDRAW | CS_VREDRAW;
    wcex.lpfnWndProc    = (WNDPROC)WndProc;
    wcex.cbClsExtra     = 0;
    wcex.cbWndExtra     = 0;
    wcex.hInstance      = hInstance;
    wcex.hIcon          = LoadIcon(hInstance, (LPCTSTR)IDI_IW98_1);
    wcex.hCursor        = LoadCursor(NULL, IDC_ARROW);
    wcex.hbrBackground  = (HBRUSH)(COLOR_WINDOW+1);
    wcex.lpszMenuName   = (LPCSTR)IDC_IW98_1;
    wcex.lpszClassName  = szWindowClass;
    wcex.hIconSm        = LoadIcon(wcex.hInstance, (LPCTSTR)IDI_SMALL);

    return RegisterClassEx( &wcex );
}

//
//   関数: InitInstance(HANDLE, int)
//
//   用途: インスタンス ハンドルの保存とメイン ウィンドウの作成
//
//
BOOL InitInstance( HINSTANCE hInstance, int nCmdShow )
{
   HWND hWnd;

   hInst = hInstance; // グローバル変数にインスタンス ハンドルを保存します

   hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
      CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

   if( !hWnd )
   {
      return FALSE;
   }

   ShowWindow( hWnd, nCmdShow );
   UpdateWindow( hWnd );

   return TRUE;
}
```

```
131
132 //
133 //  関数: WndProc(HWND, unsigned, WORD, LONG)
134 //
135 //  用途: メイン ウィンドウのメッセージを処理します。
136 //
137 //  WM_COMMAND  - アプリケーション メニューの処理
138 //  WM_PAINT    - メイン ウィンドウの描画
139 //  WM_DESTROY  - 終了メッセージの通知とリターン
140 //
141 //
142 LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
143 {
144     int wmId, wmEvent;
145     TCHAR szHello[MAX_LOADSTRING];
146     LoadString(hInst, IDS_HELLO, szHello, MAX_LOADSTRING);
147
148     switch( message )
149     {
150     case WM_COMMAND:
151         wmId    = LOWORD(wParam);
152         wmEvent = HIWORD(wParam);
153         // メニュー選択の解析:
154         switch( wmId )
155         {
156         case IDM_EXIT:
157             DestroyWindow( hWnd );
158             break;
159         case ID_SEND_MAIL_WSA:
160             DialogBox(hInst, (LPCTSTR)IDD_SENDMAIL, hWnd, (DLGPROC)SendMailProc);
161             break;
162         case ID_RECV_MAIL_WSA:
163             DialogBox(hInst, (LPCTSTR)IDD_POP, hWnd, (DLGPROC)RecvMailProc);
164             break;
165         case ID_SEND_MAIL:
166             DialogBox(hInst, (LPCTSTR)IDD_SENDMAIL, hWnd, (DLGPROC)BerkeleySendMailPro
    c);
167             break;
168         case ID_RECV_MAIL:
169             DialogBox(hInst, (LPCTSTR)IDD_POP, hWnd, (DLGPROC)BerkeleyRecvMailProc);
170             break;
171         default:
172            return DefWindowProc( hWnd, message, wParam, lParam );
173         }
174         break;
175
176     case WM_DESTROY:
177         PostQuitMessage( 0 );
178         break;
179
180     default:
181         return DefWindowProc( hWnd, message, wParam, lParam );
182    }
183    return 0;
184 }
185
```

```
 1 //
 2 //  POP sample
 3 //  Copyright (c) 1999 Orangesoft Inc.
 4 //
 5 #include "stdafx.h"
 6 #include <winsock.h>
 7 #include <string>
 8
 9 #include "resource.h"
10 #include "recvmail.h"
11
12 using namespace std;
13
14 #define UM_SELECT       (WM_USER+100)
15 #define UM_HOSTBYNAME   (WM_USER+101)
16 #define UM_QUIT         (WM_USER+102)
17
18 #define MAX_USER        1000
19 #define MAX_PASS        1000
20
21 #define POP_HOST        "hogehoge"
22 #define POP_PORT        110
23
24 enum POPSTAT {
25     ST_START,
26     ST_CONNECTTING,
27     ST_POP_GREEDING,
28     ST_POP_USER,
29     ST_POP_PASS,
30     ST_POP_STAT,
31     ST_POP_RESP_STAT,
32     ST_POP_RETR,
33     ST_POP_RESP_RETR,
34     ST_POP_RECV_MESSAGE,
35     ST_POP_QUIT,
36     ST_CLOSE
37 };
38
39 static int      g_state;
40 static char     g_bufHostent[MAXGETHOSTSTRUCT];
41 static char     g_szUser[MAX_USER];
42 static char     g_szPass[MAX_PASS];
43 static SOCKET   g_socket;
44 static char*    g_pszRecvBuffer = NULL;
45 static int      g_nBufSize = 0;
46 static int      g_nMessageNumber;
47 static int      g_nTotalCount;
48 static bool     g_bInHeader;
49
50 static bool OnOk(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
51 {
52     GetWindowText(GetDlgItem(hDlg, IDC_EDIT_USER), g_szUser, MAX_USER);
53     GetWindowText(GetDlgItem(hDlg, IDC_EDIT_PASSWORD), g_szPass, MAX_PASS);
54
55     if(strlen(g_szUser) == 0 || strlen(g_szPass) == 0){
56         return false;
57     }
58
59     return WSAAsyncGetHostByName(hDlg, UM_HOSTBYNAME, POP_HOST,
60                 g_bufHostent, MAXGETHOSTSTRUCT) != 0;
61 }
62
63 static bool OnGetHostByName(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
64 {
65     int nError = WSAGETASYNCERROR(lParam);
66     if(nError != 0){
67         return false;
68     }
69     LPHOSTENT lpHostent = (LPHOSTENT)&g_bufHostent;
70
71     g_socket = socket(AF_INET, SOCK_STREAM, 0);
72     if(g_socket == INVALID_SOCKET){
73         return false;
74     }
75
76     // 非同期モードに移行
77     if(WSAAsyncSelect(g_socket, hDlg, UM_SELECT,
78                 FD_CONNECT|FD_READ|FD_WRITE|FD_CLOSE) == SOCKET_ERROR)
79     {
80         return false;
81     }
82
83     //
84     SOCKADDR_IN sockaddr;
85     sockaddr.sin_family = AF_INET;
86     sockaddr.sin_port = htons(POP_PORT);;
87     sockaddr.sin_addr = *((LPIN_ADDR)*lpHostent->h_addr_list);
88
89     if(connect(g_socket, (LPSOCKADDR)&sockaddr, sizeof(SOCKADDR_IN)) == SOCKET_ERROR)
90     {
91         if(WSAGetLastError() != WSAEWOULDBLOCK){
92             return false;
93         }
94     }
95     g_state = ST_CONNECTTING;
96
97     return true;
98 }
99
100 static void CloseSocket()
101 {
102     closesocket(g_socket);
103     g_state = ST_CLOSE;
104 }
105
106 // コマンドの送出
107 static bool SendComand(LPCSTR lpszCommand)
108 {
109     string cmd = lpszCommand;
110     cmd += "\r\n";
111     if(!SendData(g_socket, cmd.c_str())){
112         CloseSocket();
113         return false;
114     }
115     return true;
116 }
117
118 bool ParsePopResponse(string& text)
119 {
120     if(text.substr(0, 3) == "+OK"){
121         return true;
122     }
123     return false;
124 }
125
126 // 行の解析。
127 static bool ParseLine(HWND hDlg, string& text)
128 {
129     char szCommand[256];
130
```

```
131     if(g_state != ST_POP_RECV_MESSAGE){
132         // レスポンスの受信
133         bool bResult = ParsePopResponse(text);
134         if(!bResult){
135             SendComand("QUIT");
136             g_state = ST_POP_QUIT;
137             return false;
138         }
139     }
140
141     bool bContinueLoop;
142     do{
143         bContinueLoop = false;
144         switch(g_state)
145         {
146         case ST_POP_GREEDING:
147             wsprintf(szCommand, "USER %s", g_szUser);
148             if(!SendComand(szCommand)){
149                 return false;
150             }
151             g_state = ST_POP_PASS;
152             break;
153
154         case ST_POP_PASS:
155             wsprintf(szCommand, "PASS %s", g_szPass);
156             if(!SendComand(szCommand)){
157                 return false;
158             }
159             g_state = ST_POP_STAT;
160             break;
161
162         case ST_POP_STAT:
163             if(!SendComand("STAT")){
164                 return false;
165             }
166             g_state = ST_POP_RESP_STAT;
167             break;
168
169         case ST_POP_RESP_STAT:
170             g_nTotalCount = atoi(text.c_str()+4);
171             g_nMessageNumber = 1;
172             g_state = ST_POP_RETR;
173             bContinueLoop = true;
174             break;
175
176         case ST_POP_RETR:
177             if(g_nMessageNumber > g_nTotalCount){
178                 if(!SendComand("QUIT")){
179                     return false;
180                 }
181                 g_state = ST_POP_QUIT;
182                 break;
183             }
184
185             wsprintf(szCommand, "RETR %d", g_nMessageNumber++);
186             g_bInHeader = true;
187             if(!SendComand(szCommand)){
188                 return false;
189             }
190             g_state = ST_POP_RESP_RETR;
191             break;
192
193         case ST_POP_RESP_RETR:
194             g_state = ST_POP_RECV_MESSAGE;
195             break;
196
197         case ST_POP_RECV_MESSAGE:
198             if(text == ".\r\n"){
199                 // end of message
200                 g_state = ST_POP_RETR;
201                 bContinueLoop = true;
202             }else{
203                 if(g_bInHeader){
204                     // ヘッダをリストに表示
205                     SendMessage(GetDlgItem(hDlg, IDC_LIST), LB_ADDSTRING, 0, (LPARAM)t
ext.c_str());
206                 }
207                 if(text == "\r\n"){
208                     g_bInHeader = false;
209                 }
210             }
211             break;
212
213         case ST_POP_QUIT:
214             CloseSocket();
215             PostMessage(hDlg, UM_QUIT, 0, 0);
216             break;
217         }
218     }while(bContinueLoop);
219     return true;
220 }
221
222 // 受信したデータを行単位に切り出す
223 static bool ParseData(HWND hDlg, const char* buf)
224 {
225     const int BUF_UNIT_SIZE = 2000;
226     if(g_pszRecvBuffer == NULL){
227         g_pszRecvBuffer = new char[BUF_UNIT_SIZE];
228         *g_pszRecvBuffer = '\0';
229         g_nBufSize = BUF_UNIT_SIZE;
230     }
231
232     // bufをg_pszRecvBufferに追加
233     int cnt = strlen(buf);
234     if(strlen(g_pszRecvBuffer)+cnt+1 > g_nBufSize){
235         // g_pszRecvBufferを拡大
236         int nNewBufSize = ((strlen(g_pszRecvBuffer)+cnt+1)/BUF_UNIT_SIZE+1)*BUF_UNIT_S
IZE;
237         char* pszNewBuffer = new char[nNewBufSize];
238         strcpy(pszNewBuffer, g_pszRecvBuffer);
239         delete g_pszRecvBuffer;
240         g_pszRecvBuffer = pszNewBuffer;
241         g_nBufSize = nNewBufSize;
242     }
243     strcat(g_pszRecvBuffer, buf);
244
245     // 行を切り出してParseLineに渡す。
246     while(true){
247         LPTSTR lpszPos = strchr(g_pszRecvBuffer, '\n');
248         if(lpszPos == NULL){
249             break;
250         }
251         *lpszPos = '\0';
252         string text = g_pszRecvBuffer;
253         text += "\n";
254         if(strlen(lpszPos+1) > 0){
255             memmove(g_pszRecvBuffer, lpszPos+1, strlen(lpszPos+1)+1);
256         }else{
257             *g_pszRecvBuffer = '\0';
258         }
```

```
259         if(!ParseLine(hDlg, text)){
260             return false;
261         }
262     }
263     return true;
264 }
265
266 static bool OnSelect(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
267 {
268     switch(WSAGETSELECTEVENT(lParam))
269     {
270     case FD_CONNECT:
271         g_state = ST_POP_GREEDING;
272         g_pszRecvBuffer = NULL;
273         break;
274
275     case FD_READ:
276         {
277             char buf[1000];
278             int cnt = recv(g_socket, buf, sizeof(buf)-1, 0);
279             if(cnt == SOCKET_ERROR){
280                 return false;
281             }else if(cnt == 0){
282                 // closed
283                 return false;
284             }
285             *(buf+cnt) = '\0';
286             ParseData(hDlg, buf);
287         }
288         break;
289
290     case FD_WRITE:
291         break;
292
293     case FD_CLOSE:
294         break;
295     }
296     return true;
297 }
298
299 LRESULT CALLBACK RecvMailProc( HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam )
300 {
301     switch( message )
302     {
303     case WM_INITDIALOG:
304         g_state = ST_START;
305         return TRUE;
306
307     case WM_COMMAND:
308         if( LOWORD(wParam) == IDOK ){
309             SendMessage(GetDlgItem(hDlg, IDC_LIST), LB_RESETCONTENT, 0, 0);
310             OnOk(hDlg, message, wParam, lParam);
311         }else if(LOWORD(wParam) == IDCANCEL){
312             EndDialog(hDlg, LOWORD(wParam));
313             return TRUE;
314         }
315         break;
316
317     case WM_DESTROY:
318         delete[] g_pszRecvBuffer;
319         g_pszRecvBuffer = NULL;
320         break;
321
322     case UM_HOSTBYNAME:
323         OnGetHostByName(hDlg, message, wParam, lParam);
324         break;
325
326     case UM_SELECT:
327         OnSelect(hDlg, message, wParam, lParam);
328         break;
329
330     case UM_QUIT:
331         SendMessage(GetDlgItem(hDlg, IDC_LIST), LB_ADDSTRING, 0, (LPARAM)"====");
332         return TRUE;
333     }
334     return FALSE;
335 }
336
```

```cpp
//
//  POP sample
//  Copyright (c) 1999 Orangesoft Inc.
//
#include "stdafx.h"
#include <winsock.h>
#include <process.h>

#include "resource.h"
#include "recvmail.h"

#define UM_ENDTHREAD    (WM_USER+100)

#define POP_HOST        "hogehoge"
#define POP_PORT        110

// コマンドの送出
static bool SendPopComand(SOCKET sock, LPCSTR lpszCommand)
{
    string cmd = lpszCommand;
    cmd += "\r\n";
    if(!SendData(sock, cmd.c_str())){
        closesocket(sock);
        return false;
    }
    return true;
}

bool RecvPopResponse(SOCKET sock, string& line)
{
    if(!RecvLine(sock, line)){
        return false;
    }
    return ParsePopResponse(line);
}

struct PopParam
{
    string  strUser;
    string  strPass;
    HWND    hwndNotify;
    HWND    hwndListBox;
};

bool PopProcImpl(PopParam* pParam)
{
    char szCommand[256];
    string line;

    // create
    SOCKET sock = Connect(POP_HOST, POP_PORT);

    // read greeding
    if(!RecvPopResponse(sock, line)){
        return false;
    }

    // USER
    wsprintf(szCommand, "USER %s", pParam->strUser.c_str());
    if(!SendPopComand(sock, szCommand)){
        return false;
    }
    if(!RecvPopResponse(sock, line)){
        return false;
    }

    // PASS
    wsprintf(szCommand, "PASS %s", pParam->strPass.c_str());
    if(!SendPopComand(sock, szCommand)){
        return false;
    }
    if(!RecvPopResponse(sock, line)){
        return false;
    }

    // STAT
    if(!SendPopComand(sock, "STAT")){
        return false;
    }
    if(!RecvPopResponse(sock, line)){
        return false;
    }

    int nTotalCount = atoi(line.c_str()+4);

    // retr
    for(int n=0; n<nTotalCount; ++n){
        wsprintf(szCommand, "RETR %d", n+1);
        if(!SendPopComand(sock, szCommand)){
            return false;
        }

        bool bHeader = true;
        string header;
        while(true) {
            if(!RecvLine(sock, line)){
                return false;
            }
            if(bHeader){
                SendMessage(pParam->hwndListBox, LB_ADDSTRING, 0, (LPARAM)line.c_str()
);
            }
            if(line == "\r\n"){
                bHeader = false;
            }else if(line == ".\r\n"){
                break;
            }
        }
    }

    // QUIT
    if(!SendPopComand(sock, "QUIT")){
        return false;
    }
    if(!RecvPopResponse(sock, line)){
        return false;
    }

    //
    closesocket(sock);

    return true;
}

void PopProc(void* pParam)
{
    PopParam* pPopParam = (PopParam*)pParam;
    bool bResult = PopProcImpl(pPopParam);
    PostMessage(pPopParam->hwndNotify, UM_ENDTHREAD, bResult ? TRUE : FALSE, 0);
}
```

```
130
131 static void OnOk(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
132 {
133     const int MAX_USER = 1000;
134     const int MAX_PASS = 1000;
135
136     static PopParam param;
137     char szUser[MAX_USER];
138     char szPass[MAX_PASS];
139
140     GetWindowText(GetDlgItem(hDlg, IDC_EDIT_USER), szUser, MAX_USER);
141     GetWindowText(GetDlgItem(hDlg, IDC_EDIT_PASSWORD), szPass, MAX_PASS);
142
143     if(strlen(szUser) == 0 || strlen(szPass) == 0){
144         return;
145     }
146
147     param.hwndNotify = hDlg;
148     param.hwndListBox = GetDlgItem(hDlg, IDC_LIST);
149     param.strUser = szUser;
150     param.strPass = szPass;
151
152     // start thread
153     _beginthread(PopProc, 0, (void*)&param);
154 }
155
156 LRESULT CALLBACK BerkeleyRecvMailProc( HWND hDlg, UINT message, WPARAM wParam, LPARAM
    lParam )
157 {
158     switch( message )
159     {
160     case WM_INITDIALOG:
161         return TRUE;
162
163     case WM_COMMAND:
164         if( LOWORD(wParam) == IDOK ){
165             OnOk(hDlg, message, wParam, lParam);
166         }else if(LOWORD(wParam) == IDCANCEL){
167             EndDialog(hDlg, LOWORD(wParam));
168             return TRUE;
169         }
170         break;
171
172     case UM_ENDTHREAD:
173         EndDialog(hDlg, LOWORD(wParam));
174         return TRUE;
175     }
176     return FALSE;
177 }
178
```

```cpp
//
//  SMTP sample
//  Copyright (c) 1999 Orangesoft Inc.
//

#include "stdafx.h"
#include <winsock.h>
#include <string>

#include "resource.h"
#include "sendmail.h"

using namespace std;

#define UM_SELECT       (WM_USER+100)
#define UM_HOSTBYNAME   (WM_USER+101)
#define UM_POSTEND      (WM_USER+102)

#define MAX_TO          1000
#define MAX_SUBJECT     1000
#define MAX_BODY        1000

#define SMTP_HOST       "hogehoge"
#define SMTP_PORT       25
#define FQDN            "hogehoge.co.jp"
#define MYADDRESS       "hoge@hogehoge.co.jp"

enum SMTP_STAT {
    ST_START,
    ST_CONNECTTING,
    ST_SMTP_GREEDING,
    ST_SMTP_HELO,
    ST_SMTP_MAILFROM,
    ST_SMTP_RCPT,
    ST_SMTP_DATA,
    ST_SMTP_QUIT,
    ST_CLOSE
};

static int      g_state;
static char     g_bufHostent[MAXGETHOSTSTRUCT];
static char     g_szTo[MAX_TO];
static char     g_szSubject[MAX_SUBJECT];
static char     g_szBody[MAX_BODY];
static SOCKET   g_socket;
static char*    g_pszRecvBuffer = NULL;
static int      g_nBufSize = 0;

static bool OnOk(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    GetWindowText(GetDlgItem(hDlg, IDC_EDIT_TO), g_szTo, MAX_TO);
    GetWindowText(GetDlgItem(hDlg, IDC_EDIT_SUBJECT), g_szSubject, MAX_SUBJECT);
    GetWindowText(GetDlgItem(hDlg, IDC_EDIT_BODY), g_szBody, MAX_BODY);

    if(strlen(g_szTo) == 0){
        return false;
    }

    return WSAAsyncGetHostByName(hDlg, UM_HOSTBYNAME, SMTP_HOST,
                g_bufHostent, MAXGETHOSTSTRUCT) != 0;
}

static bool OnGetHostByName(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    int nError = WSAGETASYNCERROR(lParam);
    if(nError != 0){
        return false;
    }
    LPHOSTENT lpHostent = (LPHOSTENT)&g_bufHostent;

    g_socket = socket(AF_INET, SOCK_STREAM, 0);
    if(g_socket == INVALID_SOCKET){
        return false;
    }

    // 非同期モードに移行
    if(WSAAsyncSelect(g_socket, hDlg, UM_SELECT,
                FD_CONNECT|FD_READ|FD_WRITE|FD_CLOSE) == SOCKET_ERROR)
    {
        return false;
    }

    //
    SOCKADDR_IN sockaddr;
    sockaddr.sin_family = AF_INET;
    sockaddr.sin_port = htons(SMTP_PORT);
    sockaddr.sin_addr = *((LPIN_ADDR)*lpHostent->h_addr_list);

    if(connect(g_socket, (LPSOCKADDR)&sockaddr, sizeof(SOCKADDR_IN)) == SOCKET_ERROR)
    {
        if(WSAGetLastError() != WSAEWOULDBLOCK){
            return false;
        }
    }
    g_state = ST_CONNECTTING;

    return true;
}

void ParseSmtpResponse(string& text, int& code, bool& bContinue)
{
    string num = text.substr(0, 3);
    code = atoi(num.c_str());
    bContinue = text[3] == '-';
}

static void CloseSocket()
{
    closesocket(g_socket);
    g_state = ST_CLOSE;
}

static bool SendSmtpComand(SOCKET sock, LPCSTR lpszCommand)
{
    string cmd = lpszCommand;
    cmd += "\r\n";
    if(!SendData(sock, cmd.c_str())){
        CloseSocket();
        return false;
    }
    return true;
}

static bool SendMail()
{
    string message;
    message += string("to: ") + g_szTo + "\r\n";
    message += string("subject: ") + g_szSubject + "\r\n\r\n";
    message += g_szBody;
    message += "\r\n.\r\n";
```

```
131
132     if(!SendData(g_socket, message.c_str())){
133         CloseSocket();
134         return false;
135     }
136     return true;
137 }
138
139 static bool ParseLine(HWND hDlg, string& text)
140 {
141     int code;
142     bool bContinue;
143     char szCommand[256];
144
145     ParseSmtpResponse(text, code, bContinue);
146
147     if(bContinue){
148         return true;    // 継続行がある。
149     }
150
151     int result = code/100;
152
153     switch(g_state)
154     {
155     case ST_SMTP_GREEDING:
156         if(result != 2 ){
157             SendSmtpComand(g_socket, "QUIT");
158             g_state = ST_SMTP_QUIT;
159             return false;
160         }
161
162         wsprintf(szCommand, "HELO %s", FQDN);
163         if(!SendSmtpComand(g_socket, szCommand)){
164             return false;
165         }
166         g_state = ST_SMTP_HELO;
167         break;
168
169     case ST_SMTP_HELO:
170         if(result != 2 ){
171             SendSmtpComand(g_socket, "QUIT");
172             g_state = ST_SMTP_QUIT;
173             return false;
174         }
175
176         wsprintf(szCommand, "MAIL FROM: <%s>", MYADDRESS);
177         if(!SendSmtpComand(g_socket, szCommand)){
178             return false;
179         }
180         g_state = ST_SMTP_MAILFROM;
181         break;
182
183     case ST_SMTP_MAILFROM:
184         if(result != 2 ){
185             SendSmtpComand(g_socket, "QUIT");
186             g_state = ST_SMTP_QUIT;
187             return false;
188         }
189
190         wsprintf(szCommand, "RCPT TO: <%s>", g_szTo);
191         if(!SendSmtpComand(g_socket, szCommand)){
192             return false;
193         }
194         g_state = ST_SMTP_RCPT;
195         break;
196
197     case ST_SMTP_RCPT:
198         // "2yz"以外はQUITコマンドを送出して終了。
199         if(result != 2 ){
200             SendSmtpComand(g_socket, "QUIT");
201             g_state = ST_SMTP_QUIT;
202             return false;
203         }
204
205         if(!SendSmtpComand(g_socket, "DATA")){
206             return false;
207         }
208         g_state = ST_SMTP_DATA;
209         break;
210
211     case ST_SMTP_DATA:
212         // "2yz"以外はQUITコマンドを送出して終了。
213         if(result != 3 ){
214             SendSmtpComand(g_socket, "QUIT");
215             g_state = ST_SMTP_QUIT;
216             return false;
217         }
218
219         if(!SendMail()){
220             CloseSocket();
221             return false;
222         }
223
224         if(!SendSmtpComand(g_socket, "QUIT")){
225             return false;
226         }
227         g_state = ST_SMTP_QUIT;
228         break;
229
230     case ST_SMTP_QUIT:
231         CloseSocket();
232         PostMessage(hDlg, UM_POSTEND, 0, 0);
233         break;
234     }
235
236     return true;
237 }
238
239 // 行を切り出す
240 static bool ParseData(HWND hDlg, const char* buf)
241 {
242     const int BUF_UNIT_SIZE = 2000;
243     if(g_pszRecvBuffer == NULL){
244         g_pszRecvBuffer = new char[BUF_UNIT_SIZE];
245         *g_pszRecvBuffer = '\0';
246         g_nBufSize = BUF_UNIT_SIZE;
247     }
248
249     // bufをg_pszRecvBufferに追加
250     int cnt = strlen(buf);
251     if(strlen(g_pszRecvBuffer)+cnt+1 > g_nBufSize){
252         // g_pszRecvBufferを拡大
253         int nNewBufSize = ((strlen(g_pszRecvBuffer)+cnt+1)/BUF_UNIT_SIZE+1)*BUF_UNIT_S
IZE;
254         char* pszNewBuffer = new char[nNewBufSize];
255         strcpy(pszNewBuffer, g_pszRecvBuffer);
256         delete g_pszRecvBuffer;
257         g_pszRecvBuffer = pszNewBuffer;
258         g_nBufSize = nNewBufSize;
259     }
```

```
260     strcat(g_pszRecvBuffer, buf);
261
262     // 行を切り出してParseLineに渡す。
263     while(true){
264         LPTSTR lpszPos = strchr(g_pszRecvBuffer, '\n');
265         if(lpszPos == NULL){
266             break;
267         }
268         *lpszPos = '\0';
269         string text = g_pszRecvBuffer;
270         text += "\n";
271         if(strlen(lpszPos+1) > 0){
272             memmove(g_pszRecvBuffer, lpszPos+1, strlen(lpszPos+1)+1);
273         }else{
274             *g_pszRecvBuffer = '\0';
275         }
276         if(!ParseLine(hDlg, text)){
277             return false;
278         }
279     }
280     return true;
281 }
282
283 static bool OnSelect(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
284 {
285     switch(WSAGETSELECTEVENT(lParam))
286     {
287     case FD_CONNECT:
288         g_state = ST_SMTP_GREEDING;
289         g_pszRecvBuffer = NULL;
290         break;
291
292     case FD_READ:
293         {
294             char buf[1000];
295             int cnt = recv(g_socket, buf, sizeof(buf)-1, 0);
296             if(cnt == SOCKET_ERROR){
297                 return false;
298             }else if(cnt == 0){
299                 // closed
300                 return false;
301             }
302             *(buf+cnt) = '\0';
303             ParseData(hDlg, buf);
304         }
305         break;
306
307     case FD_WRITE:
308         break;
309
310     case FD_CLOSE:
311         break;
312     }
313     return true;
314 }
315
316 LRESULT CALLBACK SendMailProc( HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam )
317 {
318     switch( message )
319     {
320     case WM_INITDIALOG:
321         g_state = ST_START;
322         return TRUE;
323
324     case WM_COMMAND:
325         if( LOWORD(wParam) == IDOK ){
326             OnOk(hDlg, message, wParam, lParam);
327         }else if(LOWORD(wParam) == IDCANCEL){
328             EndDialog(hDlg, LOWORD(wParam));
329             return TRUE;
330         }
331         break;
332
333     case WM_DESTROY:
334         delete[] g_pszRecvBuffer;
335         g_pszRecvBuffer = NULL;
336         break;
337
338     case UM_HOSTBYNAME:
339         OnGetHostByName(hDlg, message, wParam, lParam);
340         break;
341
342     case UM_SELECT:
343         OnSelect(hDlg, message, wParam, lParam);
344         break;
345
346     case UM_POSTEND:
347         EndDialog(hDlg, LOWORD(wParam));
348         return TRUE;
349     }
350     return FALSE;
351 }
352
```

```cpp
//
//  SMTP sample
//  Copyright (c) 1999 Orangesoft Inc.
//

#include "stdafx.h"
#include <winsock.h>
#include <process.h>

#include "resource.h"
#include "sendmail.h"

#define UM_ENDTHREAD    (WM_USER+100)

#define SMTP_HOST       "hogehoge"
#define SMTP_PORT       25
#define FQDN            "hogehoge.co.jp"
#define MYADDRESS       "hoge@hogehoge.co.jp"

int RecvCommandResponse(SOCKET sock)
{
    string line;
    int code;
    bool bContinue;
    do{
        if(!RecvLine(sock, line)){
            return -1;
        }
        ParseSmtpResponse(line, code, bContinue);
        if((code/100) == 2){
            return code;
        }
    }while(bContinue);
    return code;
}

struct SmtpParam
{
    string  strSubject;
    string  strTo;
    string  strBody;
    HWND    hwndNotify;
};

static bool SendSmtpComand(SOCKET sock, LPCSTR lpszCommand)
{
    string cmd = lpszCommand;
    cmd += "\r\n";
    if(!SendData(sock, cmd.c_str())){
        closesocket(sock);
        return false;
    }
    return true;
}

bool SmtpProcImpl(SmtpParam* pParam)
{
    char szCommand[256];
    int code;

    // create
    SOCKET sock = Connect(SMTP_HOST, SMTP_PORT);

    // read greeding
    code = RecvCommandResponse(sock);
    if(code < 0 || (code/100) != 2){
        return false;
    }

    // helo
    wsprintf(szCommand, "HELO %s", FQDN);
    if(!SendSmtpComand(sock, szCommand)){
        return false;
    }
    code = RecvCommandResponse(sock);
    if(code < 0 || (code/100) != 2){
        return false;
    }

    //
    wsprintf(szCommand, "MAIL FROM: <%s>", MYADDRESS);
    if(!SendSmtpComand(sock, szCommand)){
        return false;
    }
    code = RecvCommandResponse(sock);
    if(code < 0 || (code/100) != 2){
        return false;
    }

    //
    wsprintf(szCommand, "RCPT TO: <%s>", pParam->strTo.c_str());
    if(!SendSmtpComand(sock, szCommand)){
        return false;
    }
    code = RecvCommandResponse(sock);
    if(code < 0 || (code/100) != 2){
        return false;
    }

    //
    if(!SendSmtpComand(sock, "DATA")){
        return false;
    }
    code = RecvCommandResponse(sock);
    if(code < 0 || (code/100) != 3){
        return false;
    }

    //
    string message;
    message += string("to: ") + pParam->strTo + "\r\n";
    message += string("subject: ") + pParam->strSubject + "\r\n\r\n";
    message += pParam->strBody;
    message += "\r\n.\r\n";

    if(!SendData(sock, message.c_str())){
        closesocket(sock);
        return false;
    }

    //
    if(!SendSmtpComand(sock, "QUIT")){
        return false;
    }
    code = RecvCommandResponse(sock);
    if(code < 0 || (code/100) != 2){
        return false;
    }

    //
```

```
131     closesocket(sock);
132
133     return true;
134 }
135
136 void SmtpProc(void* pParam)
137 {
138     SmtpParam* pSmtpParam = (SmtpParam*)pParam;
139     bool bResult = SmtpProcImpl(pSmtpParam);
140     PostMessage(pSmtpParam->hwndNotify, UM_ENDTHREAD, bResult ? TRUE : FALSE, 0);
141 }
142
143 static void OnOk(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
144 {
145     static SmtpParam param;
146
147     const UINT BUF_MAX = 1000;
148     char szBuffer[BUF_MAX];
149     GetWindowText(GetDlgItem(hDlg, IDC_EDIT_TO), szBuffer, BUF_MAX);
150     param.strTo = szBuffer;
151     GetWindowText(GetDlgItem(hDlg, IDC_EDIT_SUBJECT), szBuffer, BUF_MAX);
152     param.strSubject = szBuffer;
153     GetWindowText(GetDlgItem(hDlg, IDC_EDIT_BODY), szBuffer, BUF_MAX);
154     param.strBody = szBuffer;
155
156     if(param.strTo.length() == 0){
157         return;
158     }
159     param.hwndNotify = hDlg;
160
161     // start thread
162     _beginthread(SmtpProc, 0, (void*)&param);
163 }
164
165 LRESULT CALLBACK BerkeleySendMailProc( HWND hDlg, UINT message, WPARAM wParam, LPARAM
    lParam )
166 {
167     switch( message )
168     {
169     case WM_INITDIALOG:
170         return TRUE;
171
172     case WM_COMMAND:
173         if( LOWORD(wParam) == IDOK ){
174             OnOk(hDlg, message, wParam, lParam);
175         }else if(LOWORD(wParam) == IDCANCEL){
176             EndDialog(hDlg, LOWORD(wParam));
177             return TRUE;
178         }
179         break;
180
181     case UM_ENDTHREAD:
182         if(wParam == TRUE){
183             EndDialog(hDlg, LOWORD(wParam));
184             return TRUE;
185         }
186         break;
187     }
188
189     return FALSE;
190 }
191
```

```
 1 //
 2 //  SMTP sample
 3 //  Copyright (c) 1999 Orangesoft Inc.
 4 //
 5
 6 #include "stdafx.h"
 7 #include <string>
 8 #include <winsock.h>
 9 #include "sockbasic.h"
10
11 using namespace std;
12
13 SOCKET Connect(LPCTSTR lpszHostName, UINT port)
14 {
15     // create
16     SOCKET sock = socket(AF_INET, SOCK_STREAM, 0);
17     if(sock == INVALID_SOCKET){
18         return INVALID_SOCKET;
19     }
20
21     // 名前の解決
22     SOCKADDR_IN addr;
23     memset(&addr, sizeof(SOCKADDR_IN), 0);
24     addr.sin_family = AF_INET;
25     addr.sin_port = htons(port);
26
27     long l = inet_addr(lpszHostName);
28     if(l == INADDR_NONE){
29         LPHOSTENT lpHostent = gethostbyname(lpszHostName);
30         if(lpHostent == NULL){
31             return INVALID_SOCKET;
32         }else{
33             memcpy(&(addr.sin_addr), lpHostent->h_addr, lpHostent->h_length);
34         }
35     }else{
36         addr.sin_addr.s_addr = l;
37     }
38
39     // connect
40     if(connect(sock, (LPSOCKADDR)&addr, sizeof(SOCKADDR_IN)) == SOCKET_ERROR){
41         return INVALID_SOCKET;
42     }
43
44     return sock;
45 }
46
47 bool SendData(SOCKET sock, LPCSTR lpszData)
48 {
49     string data = lpszData;
50
51     while(true){
52         int cnt = send(sock, data.c_str(), data.length(), 0);
53         if(cnt == SOCKET_ERROR){
54             return false;
55         }else if(cnt == data.length()){
56             break;
57         }
58         data = data.substr(cnt);
59     }
60     return true;
61 }
62
63 bool RecvLine(SOCKET sock, string& line)
64 {
65     while(true){
66         char ch;
67         int cnt = recv(sock, &ch, sizeof(char), 0);
68         if(cnt == SOCKET_ERROR){
69             return false;
70         }else if(cnt == 0){
71             return false;       // closed
72         }
73         line += ch;
74         if(ch == '\n'){
75             return true;
76         }
77     }
78 }
79
```

```
1 // stdafx.cpp : 標準インクルードファイルを含むソース ファイル
2 //               iw98_1.pch 生成されるプリコンパイル済ヘッダー
3 //               stdafx.obj 生成されるプリコンパイル済タイプ情報
4
5 #include "stdafx.h"
6
7 // TODO: STDAFX.H に含まれていて、このファイルに記述されていない
8 // ヘッダーファイルを追加してください。
```

```
 1
 2 #if !defined(AFX_IW98_1_H__CEE76366_778C_11D2_9763_00400530B261__INCLUDED_)
 3 #define AFX_IW98_1_H__CEE76366_778C_11D2_9763_00400530B261__INCLUDED_
 4
 5 #if _MSC_VER > 1000
 6 #pragma once
 7 #endif // _MSC_VER > 1000
 8
 9 #include "resource.h"
10
11
12 #endif // !defined(AFX_IW98_1_H__CEE76366_778C_11D2_9763_00400530B261__INCLUDED_)
```

```
 1 //
 2 //  Copyright (c) 1999 Orangesoft Inc.
 3 //
 4 #ifndef __RECVMAIL_H__
 5 #define __RECVMAIL_H__
 6
 7 #include "sockbasic.h"
 8
 9 LRESULT CALLBACK RecvMailProc( HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam )
   ;
10 LRESULT CALLBACK BerkeleyRecvMailProc( HWND hDlg, UINT message, WPARAM wParam, LPARAM
   lParam );
11 bool ParsePopResponse(string& text);
12
13 #endif // __RECVMAIL_H__
```

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by iw98_1.rc
//
#define IDC_MYICON                      2
#define IDD_IW98_1_DIALOG               102
#define IDD_ABOUTBOX                    103
#define IDS_APP_TITLE                   103
#define IDM_ABOUT                       104
#define IDM_EXIT                        105
#define IDS_HELLO                       106
#define IDI_IW98_1                      107
#define IDI_SMALL                       108
#define IDC_IW98_1                      109
#define IDR_MAINFRAME                   128
#define IDD_SENDMAIL                    129
#define IDD_POP                         130
#define IDC_EDIT_TO                     1000
#define IDC_EDIT_BODY                   1001
#define IDC_EDIT_SUBJECT                1002
#define IDC_EDIT_USER                   1003
#define IDC_EDIT_PASSWORD               1004
#define IDC_LIST                        1005
#define ID_SEND_MAIL_WSA                32771
#define ID_RECV_MAIL                    32772
#define ID_SEND_MAIL                    32773
#define ID_RECV_MAIL_WSA                32774
#define IDC_STATIC                      -1

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        131
#define _APS_NEXT_COMMAND_VALUE         32775
#define _APS_NEXT_CONTROL_VALUE         1006
#define _APS_NEXT_SYMED_VALUE           110
#endif
#endif
```

```
 1 //
 2 //  Copyright (c) 1999 Orangesoft Inc.
 3 //
 4
 5 #ifndef __SENDMAIL_H__
 6 #define __SENDMAIL_H__
 7
 8 #include <string>
 9 using namespace std;
10 #include "sockbasic.h"
11
12 LRESULT CALLBACK SendMailProc( HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam )
   ;
13 LRESULT CALLBACK BerkeleySendMailProc( HWND hDlg, UINT message, WPARAM wParam, LPARAM
   lParam );
14 void ParseSmtpResponse(string& text, int& code, bool& bContinue);
15
16 #endif // __SENDMAIL_H__
```

```
 1 //
 2 //  Copyright (c) 1999 Orangesoft Inc.
 3 //
 4
 5 #ifndef __SOCKBASIC_H__
 6 #define __SOCKBASIC_H__
 7
 8 #include <string>
 9 using namespace std;
10
11 SOCKET Connect(LPCTSTR lpszHostName, UINT port);
12 bool SendData(SOCKET sock, LPCSTR lpszData);
13 bool RecvLine(SOCKET sock, string& line);
14
15 #endif // __SOCKBASIC_H__
```

```
// stdafx.h : 標準のシステム インクルード ファイル、
//            または参照回数が多く、かつあまり変更されない
//            プロジェクト専用のインクルード ファイルを記述します。
//

#if !defined(AFX_STDAFX_H__A9DB83DB_A9FD_11D0_BFD1_444553540000__INCLUDED_)
#define AFX_STDAFX_H__A9DB83DB_A9FD_11D0_BFD1_444553540000__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define WIN32_LEAN_AND_MEAN     // Windows ヘッダーから殆ど使用されないスタッフを除外します


// Windows ヘッダー ファイル:
#include <windows.h>

// C ランタイム ヘッダー ファイル
#include <stdlib.h>
#include <malloc.h>
#include <memory.h>
#include <tchar.h>

// ローカル ヘッダー ファイル

// TODO: プログラムで必要なヘッダー参照を追加してください。

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ は前行の直前に追加の宣言を挿入します。

#endif // !defined(AFX_STDAFX_H__A9DB83DB_A9FD_11D0_BFD1_444553540000__INCLUDED_)
```

```
 1 package InternetWeekSample;
 2
 3 public class IW98Sample {
 4     public static void main(java.lang.String[] args) {
 5         int protocol = 1;    // 0=POP, 1=SMTP
 6         if(args.length > 0 && args[0].equals("POP")){
 7             protocol = 0;
 8         }
 9
10         if(protocol == 1){
11             //
12             // SMTP
13             //
14             String receipts[] = new String[1];
15             receipts[0] = "hoge@orangesoft.co.jp";
16
17             jp.co.orangesoft.net.Smtp smtp = new jp.co.orangesoft.net.Smtp();
18
19             try{
20                 smtp.setFrom("hogehoge@hoge.co.jp");
21                 smtp.setFqdn("hoge.co.jp");
22                 smtp.setHostname("hoge.hoge.co.jp");
23                 smtp.setReceipts(receipts);
24                 smtp.setMessage("subject: text\r\n\r\ntest\r\n");
25                 smtp.start();
26             }catch(Exception e){
27                 System.out.println(e.getMessage());
28             }
29         }else{
30             //
31             // POP
32             //
33             jp.co.orangesoft.net.Pop pop = new jp.co.orangesoft.net.Pop();
34             try{
35                 pop.setHostname("hoge.hoge.co.jp");
36                 pop.setUser("hoge");
37                 pop.setPassword("password");
38                 pop.start();
39             }catch(Exception e){
40                 System.out.println(e.getMessage());
41             }
42         }
43     }
44 }
45
46
```

```java
package jp.co.orangesoft.net;

import java.net.*;
import java.io.*;

public class Pop extends Thread {

    public static final int DEFAULT_PORT = 110;
    static final String CRLF = "\r\n";

    private String hostname;
    private String user;
    private String password;
    private int port;

    private String encoding = "JIS";
    private Socket socket;
    private BufferedInputStream in;
    private BufferedOutputStream out;

    public Pop() {
        super();
        reset();
    }

    public void run() {
        try{
            String command = new String();

            socket = new Socket(hostname, port);
            in = new BufferedInputStream(socket.getInputStream());
            out = new BufferedOutputStream(socket.getOutputStream());

            // read greeding
            readResponse();


            // USER
            execCommand("USER "+user);

            // PASS
            execCommand("PASS "+password);

            // STAT
            String response = execCommand("STAT");
            java.util.StringTokenizer st = new java.util.StringTokenizer(response);
            if(st.countTokens() < 2){
                throw new PopException(response);
            }
            st.nextToken(); // STAT
            int nTotalCount = Integer.parseInt(st.nextToken());

            // retr
            for(int n=0; n<nTotalCount; ++n){
                writeCommand("RETR "+String.valueOf(n+1));

                boolean bHeader = true;
                String header = new String();
                while(true) {
                    String line = readLine();
                    if(bHeader){
                        System.out.print(line);
                    }
                    if(line.equals("\r\n")){
                        bHeader = false;
                    }else if(line.equals(".\r\n")){
                        break;
                    }
                }
            }

            // QUIT
            execCommand("QUIT");

        }catch(Exception e){
            System.out.println(e.getMessage());
        }
    }

    protected void finalize() throws IOException {
        socket.close();
    }

    public void reset() {
        hostname = null;
        password = null;
        port = DEFAULT_PORT;
    }

    public void setPort(int port) {
        this.port = port;
    }

    public void setHostname(String hostname) {
        this.hostname = hostname;
    }

    public void setUser(String user) {
        this.user = user;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    private void write(String text) throws IOException {
        byte[] bytes = text.getBytes(encoding);
        out.write(bytes);
    }

    private String readLine() throws IOException {
        StringBuffer buffer = new StringBuffer();
        int ch;
        while((ch = in.read()) != -1){
            buffer.append((char)ch);
            if(ch == '\n'){
                break;
            }
        }
        return buffer.toString();
    }

    private void writeCommand(String command) throws IOException {
        command += CRLF;
        write(command);
        out.flush();
    }

    private String execCommand(String command) throws IOException,PopException {
        writeCommand(command);
```

```
131         return readResponse();
132     }
133 
134     private String readResponse() throws IOException, PopException {
135         try{
136             String line = readLine();
137             if(!line.substring(0, 3).equals("+OK")){
138                 throw new PopException(line);
139             }
140             return line;
141         }catch(StringIndexOutOfBoundsException e){
142             throw new IOException();    // 読み込んだ文字列が少ないときなどに発生
143         }
144     }
145 
146 }
```

```
 1 package jp.co.orangesoft.net;
 2
 3 public class PopException extends Exception {
 4     String errorMessage;
 5     public PopException() {
 6     }
 7     public PopException(String errmsg) {
 8         errorMessage = errmsg;
 9     }
10     public String getErrorMessage() {
11         return errorMessage;
12     }
13 }
```

```java
package jp.co.orangesoft.net;

import java.net.*;
import java.io.*;

public class Smtp extends Thread {

    public static final int DEFAULT_PORT = 25;
    static final String CRLF = "\r\n";

    private String from;
    private String fqdn;
    private String hostname;
    private String message;
    private int port;
    private String[] receipts;

    private String encoding = "JIS";
    private Socket socket;
    private BufferedReader reader;
    private BufferedOutputStream out;

    public Smtp() {
        super();
        reset();
    }

    public void run() {
        try{
            connect();
            sendMessage();
        }catch(Exception e){
            System.out.println(e.getMessage());
        }
    }

    protected void finalize() throws IOException {
        socket.close();
    }

    public void connect() throws IOException,SmtpException {
        socket = new Socket(hostname, port);
        reader = new BufferedReader(new InputStreamReader(socket.getInputStream(), encoding));
        out = new BufferedOutputStream(socket.getOutputStream());

        // Greeding messageを読み込む
        readResponse();

        // HELO
        execCommand((fqdn != null) ? "HELO "+fqdn : "HELO");
    }

    private String readLine() throws IOException {
        try{
            String line;
            do{
                line = reader.readLine();
                System.out.println(line);
            }while(line.charAt(3) == '-');
            return line;
        }catch(StringIndexOutOfBoundsException e){
            throw new IOException();    // 読み込んだ文字列が少ないときなどに発生
        }
    }

    public void reset() {
        from = null;
        fqdn = null;
        hostname = null;
        message = null;
        port = DEFAULT_PORT;
        receipts = null;
    }

    public void sendMessage() throws IOException,SmtpException {
        execCommand("RSET");
        execCommand("MAIL FROM: <" + from + ">");

        // REPT TO ...
        for(int n=0; n<receipts.length; ++n){
            execCommand("RCPT TO: <"+receipts[n]+">");
        }

        execCommand("DATA");
        write(message);
        execCommand(".");
        out.flush();
    }

    public void setFqdn(String fqdn) {
        this.fqdn = fqdn;
    }

    public void setFrom(String from) {
        this.from = from;
    }

    public void setHostname(String hostname) {
        this.hostname = hostname;
    }

    public void setMessage(String message) {
        if(message.length() > 2){
            if(!message.substring(message.length()-2, message.length()).equals(CRLF)){
                message += CRLF;
            }
        }else{
            message += CRLF;
        }
        this.message = message;
    }

    public void setPort(int port) {
        this.port = port;
    }

    public void setReceipts(String[] receipts) {
        this.receipts = receipts;
    }

    private void write(String text) throws IOException {
        byte[] bytes = text.getBytes(encoding);
        out.write(bytes);
    }

    private void execCommand(String command) throws IOException,SmtpException {
        command += CRLF;
        write(command);
        out.flush();
```

```
130         readResponse();
131     }
132
133     private void readResponse() throws IOException, SmtpException {
134         try{
135             String line = readLine();
136             switch(line.charAt(0))
137             {
138             case '1':       // 正常な準備段階
139             case '2':       // 正常終了
140             case '3':       // 正常継続
141                 break;
142             case '4':       // 一時的な異常終了
143             case '5':       // 異常終了
144                 throw new SmtpException(line);
145             }
146         }catch(StringIndexOutOfBoundsException e){
147             throw new IOException();    // 読み込んだ文字列が少ないときなどに発生
148         }
149     }
150
151 /*  private boolean writeMessage(String message) throws IOException {
152         write(message);
153         write(CRLF+"."+CRLF);
154         out.flush();
155         return true;
156     }*/
157 }
```

```
 1 package jp.co.orangesoft.net;
 2
 3 public class SmtpException extends Exception {
 4     String errorMessage;
 5     public SmtpException() {
 6     }
 7     public SmtpException(String errmsg) {
 8         errorMessage = errmsg;
 9     }
10     public String getErrorMessage() {
11         return errorMessage;
12     }
13 }
```