2006年度プログラミング演習３
クライアント／サーバ課題
補足資料：クライアントプログラムの作成手順

ここでは，ソケット機能を使ったクライアントプログラムの作成手順について説明する．クライアントプログラムの処理手順は以下のようになる．

1. socket()によるソケット生成.
2. bzero()による sockaddr_un 構造体の初期化
3. sockaddr_un 構造体の設定
4. connect()によるサーバへの接続.
5. read() / write()によるサーバとのデータ送受信.
6. close()によるソケットのクローズ.

以下ではステップ 4-1 のプロセス１のサンプルプログラムを基に上記の手順について解説を行う．

```
/*************************************************************/
/* Step4-1_Proc-1.c：同一計算機上でのメッセージの送受　プロセス１ */
/*************************************************************/

#include <stdio.h>      // fgets(), printf(), stdin()
#include <string.h>     // strcpy(), strlen()
#include <sys/socket.h> // socket(), connect()
#include <sys/types.h>  // socket(), connect()
#include <sys/un.h>     // struct sockaddr_un

#define MAXLENGTH 256

#define SOCKETNAME "/tmp/socket" // 通信に使うソケットファイル
                                 // サーバーと同じファイルを指定する必要がある
```

・ **#define SOCKETNAME "/tmp/socket"**
ソケットを用いた PF_UNIX（UNIX ドメイン，ストリーム型ソケット）による通信では，ファイルシステムにソケットファイルを作成し，そのファイルを介して通信を行う．ソケットファイルは，サーバとクライアントで共通のファイルを使用する必要がある．

```
int main(int argc, char* argv[])
{
  char   strLine[MAXLENGTH];
  int    iFdSock;
  int    iLength;
```

```
int    iReturn;
struct sockaddr_un sockAddress;
```

・**struct sockaddr_un sockAddress**

sockaddr_un 構造体は，UNIX LOCAL なソケットのアドレスの情報を表す構造体であり，以下のメンバで構成される．

```
struct sockaddr_un {
    u_char sun_len;          /* 構造体のサイズ */
    u_char sun_family;       /* ソケットファミリーネーム */
                             /* Step4-1 では PF_UNIX を指定 */
    char   sun_path[104];    /* ソケットファイルのパスを指定 */
};
```

```
iReturn = MAXLENGTH;

// socket の作成．UNIX ドメイン，ストリーム型ソケット
if((iFdSock = socket(PF_UNIX, SOCK_STREAM, 0)) == -1){
  printf("Socket error.¥n");
  exit(-1);
}
```

1．socket()によるソケット生成

・**iFdSock = socket(PF_UNIX, SOCK_STREAM, 0);**

PF_UNIX，SOCKSTREM 型のソケットを生成する．戻り値 iFdSock はソケットの識別子である．以降の通信にはこのソケット識別子を用いる．第一引数 PF_UNIX は，ローカル通信を行うことために指定するソケットファミリーネームを表す．第二引数 SOCK_STREAM は，順序・信頼性付きの双方向バイトストリームを使用することを表す．第三引数 0 は，ソケット固有のプロトコルを使用することを表す．

```
// sockaddr_un 構造体を 0 で初期化
bzero((char*)&sockAddress, sizeof(sockAddress));
```

2．bzero()による sockaddr_un 構造体の初期化

・**bzero((char*)&sockAddress, sizeof(sockAddress));**

ソケットプログラムでは bzero()関数を用いて sckaddr_un 構造体を 0 で初期化するのがお約束である．

```
// socket の名前を設定
sockAddress.sun_family = AF_UNIX;
strcpy(sockAddress.sun_path, SOCKETNAME);
```

3. sockaddr_un 構造体の設定

- **sockAddress.sun_family = AF_UNIX;**
  ソケットを用いたローカル通信では，sockaddr_un 構造体のメンバ sun_family に AF_UNIX を指定するのがお約束である．

- **strcpy(sockAddress.sun_path, SOCKETNAME)**
  sockaddr_un 構造体のメンバ sun_path にソケットファイルのパスを設定する．

```
// サーバとの接続
// サーバ側で bind(), listen()が行われている必要がある
if(connect(iFdSock,
        (struct sockaddr*)&sockAddress,
         sizeof(sockAddress.sun_family) + strlen(SOCKETNAME)) == -1){
  printf("Connect error.¥n");
  exit(-1);
}
```

4. connect()によるサーバへの接続.

- **connect(iFdSock,(struct sockaddr*)&sockAddress,**
  sizeof(sockAddress.sun_family) + strlen(SOCKETNAME))
  iFdSock の識別子を持つソケットを sockAddress で指定したサーバに接続する．ただし．サーバ側で bind(), listen()が事前に実行されている（＝クライアントプログラムを実行する前に，サーバプログラムを実行する）必要がある．戻り値が 0 なら成功を，-1 なら失敗を表す. 第一引数 iFdScok は接続を行いたいソケットのソケット識別子を表す. 第二引数 scokAddress は接続したいサーバのアドレス等を設定した sockaddr_un 構造体を表す． 第三引数 sizeof(sockAddress.sun_family) + strlne(SOCKETNAME)は sockAddress の大きさを表す.

```
  // 入力文字列を socket に書き込んでサーバに送る
  while(fgets(strLine, MAXLENGTH, stdin) != NULL){
    write(iFdSock, strLine, MAXLENGTH);
  }
```

5. read() / write()によるサーバとのデータ送受信.

・**write(iFdSock, strLine, MAXLENGTH);**
write()関数によりサーバへメッセージを送る．第一引数 iFdSock はソケット識別子を表す．第二引数 strLine は送信用のバッファを表す．第三引数 MAXLENGTH は送信用バッファ strLine のサイズを表す．なおステップ 4-1 のプロセス 1 については，メッセージの受信は行わないため read()は使用しない．

```
  close(iFdSock);
```

6. close()によるソケットのクローズ.

・**close(iFdSock);**
socket()によって作成したソケットは必ず close()で閉じる必要がある．これは，fopen()で開いたファイルを fclose()で閉じるのと同じ概念である．

```
  return 0;
}
```