情報実験 I
三好担当分 第1回

芝浦工業大学
システム理工学部
電子情報システム学科
三好　匠

E-mail：miyoshi@shibaura-it.ac.jp
研究室HP：http://www.minet.se.shibaura-it.ac.jp/
授業支援システム：https://lms.sayo.se.shibaura-it.ac.jp/

注意

- C言語を使用します
- UNIXのソケット通信を利用します
- OSによってソケット通信の実装が異なります
- Linuxを起動して下さい
  （またはLinuxマシンにログインして下さい）

ネットワークプログラミング概要

- 担当：三好
- 全3回
  - ◆第1回：6月21日　ソケット通信
  - ◆第2回：6月28日　プロトコル（クライアント）
  - ◆第3回：7月5日　大好評！応用課題

今日の授業

- ソケット通信の実現
  - ◆プロセス間通信
  - ◆TCP/IP通信
- 授業の流れ
  - ◆ソケット通信について
  - ◆プロセス間通信の実現方法（例題）
  - ◆TCP/IP通信の実現方法（課題）

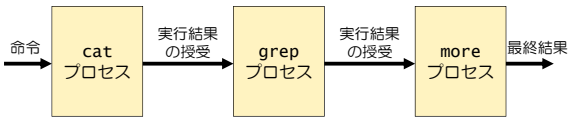
ソケット通信

プロセスとプロセス間通信

- プロセス
  - ◆個々のプログラム
  - ◆マルチタスクOSでは複数のプログラムの同時実行が可能
  - ◆1台のマシンに同時に複数のプロセスが存在
- プロセス間通信
  - ◆異なるプロセスの間でデータのやりとりを行うこと

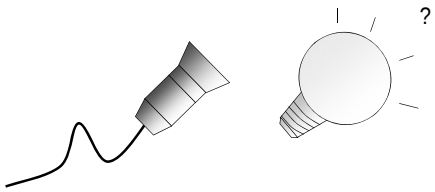
プロセス間通信の例

- UNIXシェルのパイプ処理“ | ”はプロセス間通信の一種
  - ◆`ls -l | less`
  - ◆`cat .cshrc | grep setenv | more`



自由なプロセス間通信の実現

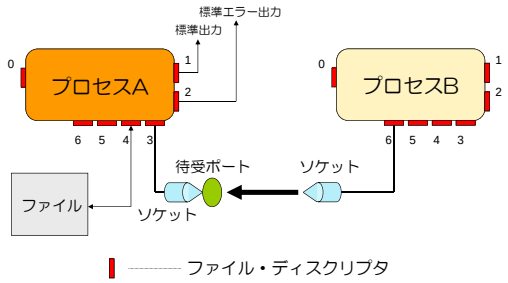
- 「ソケット」を利用したプロセス間通信
- ソケット？



ソケット

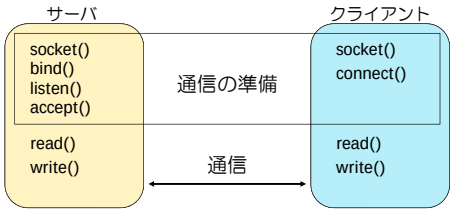
- ソケット
  - ◆プロセスとプロセスをつなぐ「通信路」の端点
- ソケットを利用してプロセス間の通信を実現
  - ◆規格化された通信が可能
  - ◆ソケットに接続して読み（受信）／書き（送信）
  - ◆通信路の実体を知らなくてもよい

ソケット通信



ソケット通信の手順

- ソケット通信を実現するにはサーバとクライアントが必要



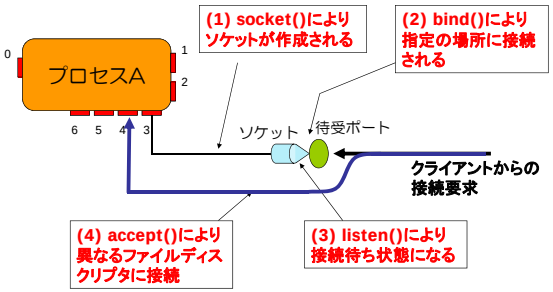
サーバの手順（1/2）

- ソケットを作る
  - ◆socket()関数の使用
  - ◆どのような種類のソケットか（プロトコルファミリ）
  - ◆どういうデータを扱うか（ソケットの型）
- ソケットをバインド（外の世界と接続）する
  - ◆bind()関数の使用
  - ◆プロトコルファミリ，その他の情報
  - ◆struct sockaddr構造体の使用

サーバの手順（2/2）

- クライアントからの接続を待つ
  - ◆listen()関数の使用
  - ◆クライアントからのアクセスをいくつ許可するか
- 接続を許可する
  - ◆accept()関数の使用
  - ◆クライアントからのアクセスを許可
  - ◆新たなソケットが作成される

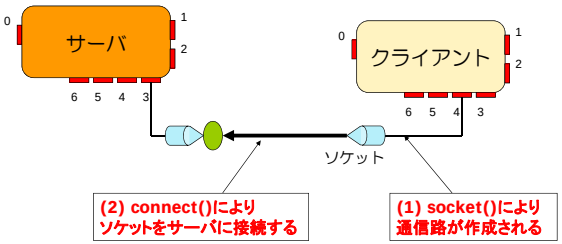
サーバの動作イメージ



クライアントの手順

- ソケットを作る
  - ◆socket()関数の使用
  - ◆どのような種類のソケットか（プロトコルファミリ）
  - ◆どういうデータを扱うか（ソケットの型）
- サーバに接続する
  - ◆connect()関数の使用
  - ◆プロトコルファミリ，その他の情報
  - ◆struct sockaddr構造体の使用

クライアントの動作イメージ



socket()関数

```
#include <sys/socket.h>
int socket(int family, int type, int protocol);
```

- **第1引数 ・・・ プロトコルファミリ**
  - ◆ AF_UNIX ・・・ UNIXプロセス間通信
  - ◆ AF_INET ・・・ IPv4プロトコル
- **第2引数 ・・・ ソケットの型**
  - ◆ SOCK_STREAM ・・・ ストリームソケット
  - ◆ SOCK_DGRAM ・・・ データグラムソケット
- **第3引数 ・・・ プロトコル番号**
  - ◆ SOCK_STREAM，SOCK_DGRAMの場合は0を指定
- **返り値 ・・・ ファイルディスクリプタ**
  - ◆ 成功なら3以上の整数，エラーなら-1

補：BSDの場合には #include <sys/types.h> を追加

bind()関数

```
#include <sys/socket.h>
int bind(int fd, struct sockaddr *addr, socklen_t len);
```

- **第1引数 ・・・ ファイルディスクリプタ**
  - ◆ socket()の返り値を使用
- **第2引数 ・・・ ソケットアドレス構造体へのポインタ**
  - ◆ struct sockaddr型の構造体を使用
- **第3引数 ・・・ソケットアドレス構造体のサイズ**
  - ◆ 第2引数で渡す構造体のサイズ
- **返り値 ・・・ bindに成功したかどうか**
  - ◆ 成功すれば0，エラーなら-1

補：BSDの場合には #include <sys/types.h> を追加

listen()関数

```
#include <sys/socket.h>
int listen(int fd, int wait);
```

- **第1引数 ・・・ ファイルディスクリプタ**
  - socket()の返り値を使用
- **第2引数 ・・・ コネクションを受け付ける数**
  - ソケットは複数のコネクションを受け付けることが可能
  - ここでは1でよい
- **返り値 ・・・ listenに成功したかどうか**
  - 成功すれば0，エラーなら-1

補：BSDの場合には #include <sys/types.h> を追加

accept()関数

```
#include <sys/socket.h>
int accept(int fd, struct sockaddr *caddr, socklen_t *len);
```

- **第1引数 ・・・ ファイルディスクリプタ**
  - socket()の返り値を使用
- **第2引数 ・・・ソケットアドレス構造体へのポインタ**
  - 接続をしてきたクライアント側のソケットアドレス構造体
- **第3引数 ・・・ソケットアドレス構造体のサイズを格納する変数へのポインタ**
  - 接続をしてきたクライアント側のソケットアドレス構造体のサイズ
- **返り値 ・・・ファイルディスクリプタ**
  - 成功なら3以上の整数，エラーなら-1

補：BSDの場合には #include <sys/types.h> を追加

connect()関数

```
#include <sys/socket.h>
int connect(int fd, struct sockaddr *saddr, socklen_t len);
```

- **第1引数 ・・・ ファイルディスクリプタ**
  - socket()の返り値を使用
- **第2引数 ・・・ソケットアドレス構造体へのポインタ**
  - 接続するサーバのソケットアドレスを記述した構造体
- **第3引数 ・・・ソケットアドレス構造体のサイズ**
  - 第2引数のサイズ
- **返り値 ・・・ connectに成功したかどうか**
  - 成功すれば0，エラーなら-1

補：BSDの場合には #include <sys/types.h> を追加

ソケットアドレス構造体

- ソケットの接続先を特定するための構造体
  - struct sockaddrで定義される
  - 構造体に格納されるデータ
    - プロトコルの種類
    - ソケットファイル，IPアドレスなど
- プロトコルの種類によって異なる構造体を使用
  - プロセス間通信：struct sockaddr_un
  - TCP/IP通信：struct sockaddr_in

通信開始後

- データを送信する
  - write()関数の使用
  - 文字列に格納されたデータを送信
- データを受信する
  - read()関数の使用
  - 送られてきたデータを文字列に格納

write()関数

```
#include <sys/unistd.h>
int write(int fd, char *buf, size_t len);
```

- **第1引数 ・・・ ファイルディスクリプタ**
  - 出力するファイルディスクリプタを指定
- **第2引数 ・・・文字列変数へのポインタ**
  - 出力したい文字列へのポインタを指定
- **第3引数 ・・・文字列の長さ**
  - 指定した文字数分だけ出力
- **返り値 ・・・出力成功した文字数**
  - エラーなら-1

read()関数

```
#include <sys/unistd.h>
int read(int fd, char *buf, size_t len);
```

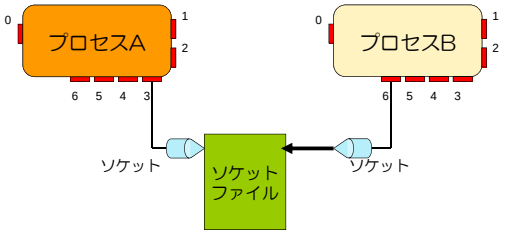
- **第1引数 ・・・ ファイルディスクリプタ**
  - 読み込むファイルディスクリプタを指定
- **第2引数 ・・・文字列変数へのポインタ**
  - 読み込んだ文字列を格納する文字列変数へのポインタを指定
- **第3引数 ・・・文字列の長さ**
  - 指定した文字数分だけ読み込む
- **返り値 ・・・読み込んだ文字数**
  - エラーなら-1

プロセス間通信

プロセス間通信

- プロセス間通信
  - 異なるプロセスの間でデータのやりとりを行うこと
  - 同一マシン上で実行された異なるプログラム間でのデータのやりとり
- ファイルを介した通信
  - UNIXでは何でもファイルとして扱う特徴
  - プロセス間でもファイルを介して通信を実現
  - ソケットをファイルにバインド（接続）

プロセス間通信のイメージ



ソケット通信のパラメータ

- プロセス間通信でのソケットの特性
  - ◆プロトコルファミリ：AF_UNIX
  - ◆ソケットの型：SOCK_STREAM
- ソケットアドレス構造体
  - ◆プロセス間通信用のソケットアドレス構造体を使用
  - ◆struct sockaddr_un
  - ◆#include <sys/un.h>
  - ◆ソケットファイル名を指定可能

struct sockaddr_unのメンバ

- sa_family_t sun_family;
  - ◆プロトコルファミリを記憶
  - ◆プロセス間通信ではAF_UNIXを指定する
- char sun_path[108];
  - ◆ソケットファイル名を指定

補:4.4BSDでは sun_path[104]

ソケットアドレス構造体の例

```
// 構造体の定義
struct sockaddr_un saddr;

// 構造体を0で初期化
memset((char *)&saddr, 0, sizeof(saddr));

// 構造体に値を代入
saddr.sun_family=AF_UNIX;
strcpy(saddr.sun_path, "./socket_file");
```


TCP/IP通信

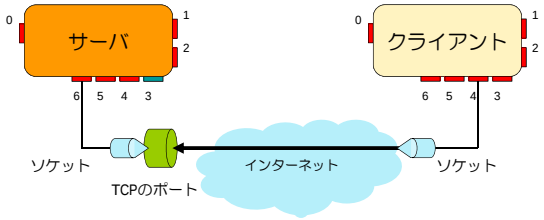
TCP/IP通信

- TCP/IP通信
  - ◆異なるマシン上で実行された異なるプログラム間でのデータのやりとり
- ソケット通信
  - ◆異なるマシン間でもソケット通信が可能
  - ◆どのマシンのどのソケットへ接続するかを指定
  - ◆ソケットをTCPのポートにバインド（接続）

TCPとポート

- TCP
  - ◆OSIの第4層であるトランスポート層のプロトコル
  - ◆任意のマシンとコネクションを確立
- ポート
  - ◆どのプロセスとの通信かを識別するための番号
  - ◆http＝80，pop3＝110
  - ◆well-known port（＜1024）

TCP/IP通信のイメージ



ソケット通信のパラメータ

- TCP/IP通信でのソケットの特性
  - ◆プロトコルファミリ：AF_INET
  - ◆ソケットの型：SOCK_STREAM
- ソケットアドレス構造体
  - ◆TCP/IP通信用のソケットアドレス構造体を使用
  - ◆struct sockaddr_in
  - ◆#include <arpa/inet.h>
  - ◆IPアドレスとポート番号を指定可能

struct sockaddr_inのメンバ

- sa_family_t sin_family;
  - ◆TCP/IP通信ではAF_INETを指定する
- in_port_t sin_port;
  - ◆TCPのポート番号を指定（Big Endianで記憶）
- struct in_addr sin_addr;
  - ◆IPアドレスを格納するための構造体

struct sockaddr_inのメンバ

- char sin_zero[8];
  - ◆未使用（構造体のサイズを調整するためのもの）
- struct in_addrの中身
  - ◆u_int32_t s_addr;
  - ◆IPアドレスを記憶する変数

ソケットアドレス構造体の例

```
// 構造体の定義
struct sockaddr_in saddr;

// 構造体を0で初期化
memset((char *)&saddr, 0, sizeof(saddr));

// 構造体に値を代入
saddr.sin_family=AF_INET;
saddr.sin_addr.s_addr=INADDR_ANY;
saddr.sin_port=htons(1357);
```


ポート番号の指定方法

- struct sockaddr_in
  - ◆sin_portにて指定
  - ◆Big Endian方式で格納
- Big Endian方式で格納するには
  - ◆htons()関数を使用
  - ◆#include <arpa/inet.h>
  - ◆saddr.sin_port=htons(9000);

補：他のシステムでは，#include <netinet/in.h>
を追加する必要があるかもしれない

IPアドレスの指定方法

- サーバ側
  - ◆任意のクライアントからの接続を許可
  - ◆INADDR_ANYを指定
  - ◆saddr.sin_addr.s_addr=INADDR_ANY;
- クライアント側
  - ◆サーバのアドレスを設定
  - ◆saddr.sin_addr.s_addr=
    inet_addr("192.168.1.1");
  - ◆#include <arpa/inet.h>

ホスト名からIPアドレスを得る方法

- マシン名→IPアドレスの変換
  - ◆gethostbyname()を利用
  - ◆#include <netdb.h>
  - ◆返り値はstruct hostent*型
- 使い方の例

```
struct hostent *hp;
if((hp=gethostbyname("www.shibaura-it.ac.jp" ))==NULL){
    perror("connect");
    exit(1);
}
memcpy(&addr.sin_addr, hp->h_addr, hp->h_length);
```


課題１

- 次の要件を満たすTCP/IP通信型大文字・小文字変換プログラム（サーバ・クライアント両方）を作成せよ
  - ◆サーバ
    - ●クライアントから送信された文字列に対し，大文字は小文字に，小文字は大文字に変換して返す
    - ●それ以外の文字は変換しない
  - ◆クライアント
    - ●キーボードから文字列を入力
    - ●入力された文字列をサーバに送信
    - ●サーバから送信された文字列を画面に出力

課題２

- 次の要件を満たすTCP/IP通信型辞書検索プログラム（サーバ・クライアント両方）を作成せよ
  - ◆サーバ
    - ●クライアントから送信された英単語に対し，その単語に対応する日本語を探索して返す
    - ●辞書にない単語の場合は，「未登録」を返す
  - ◆クライアント
    - ●キーボードから英単語を入力
    - ●入力された文字列をサーバに送信
    - ●サーバから送信された文字列を画面に出力

プログラムを作成するにあたって

- 条件（採点のため必ず守って下さい）
  - ◆クライアントではキーボードから文字列を入力させること
  - ◆クライアントはサーバに対し文字列のみを送信（write）すること
  - ◆クライアントはサーバから受信（read）した文字列（日本語）を画面表示すること
  - ◆サーバは，クライアントから文字列（英単語）を受信（read）した後，それに対応する文字列（日本語）のみを送信（write）すること．ただし，辞書にない場合には，その旨を文章として送信すること
  - ◆Writeする際には，送信する文字列の長さ（NULL文字を含む）を計算して指定すること
  - ◆ポート番号は任意に定めてよい（ただし1024以上）
  - ◆辞書ファイルはdic-w.txtを使用すること

提出方法

- 授業支援システム「Scomb」を利用
  - ◆**提出期限：6月28日13時10分**
  - ◆プログラム（サーバ×2，クライアント×1），実行結果（クライアント側×2）を提出
- プログラムの実行による採点
  - ◆次週の情報実験Ⅰの授業開始直後にプログラムを実行し，動作チェックによる採点を行う
  - ◆提出プログラムと同じものをとっておくこと

参考：IPアドレスの調べ方

- 「nslookup」コマンドを使う
  - ◆ nslookupを実行し，プロンプトがでたらマシン名を入力
  - ◆ マシン名はモニタの上部にシールで貼られています
- 実行例

```
[miyoshi@oli001 ~]$ nslookup
> oli001
Server:         202.18.117.8
Address:        202.18.117.8#53

Name:   oli001.sic.shibaura-it.ac.jp
Address: 172.24.192.111
```

```
[miyoshi@oli001 ~]$ nslookup
> sc202
Server:         202.18.117.8
Address:        202.18.117.8#53

Name:   sc202.sic.shibaura-it.ac.jp
Address: 172.24.213.202
```

- 1台のマシン内で実験する場合には，IPアドレスを127.0.0.1（ローカルアドレス）に設定しても接続できる