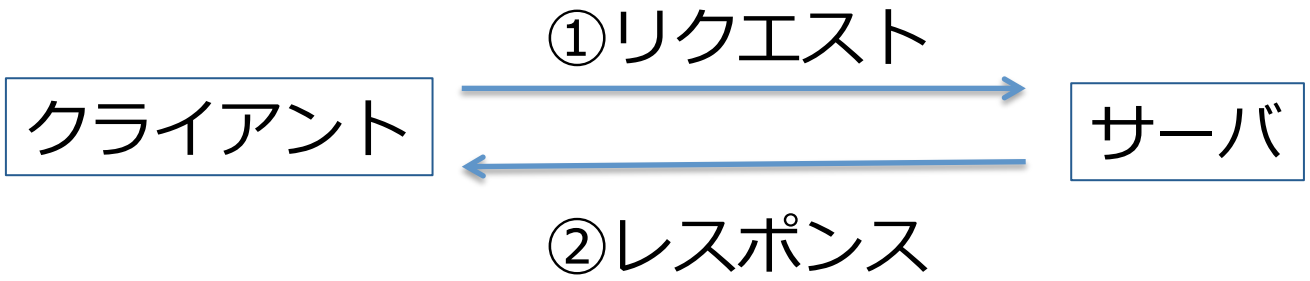
プログラミング応用
http://bit.ly/kosen02

Week2@後期
2016/09/27

本日の内容

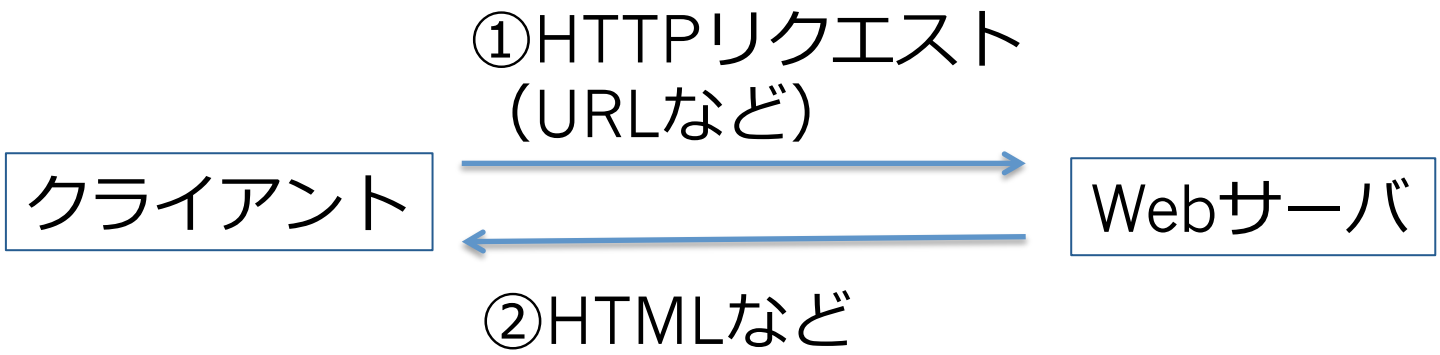
- 講義
  - クライアント・サーバ方式
  - C言語によるソケットプログラミング
    - 全体の流れ
    - よく使う関数
- 演習
  - サーバ作成
  - クライアント作成

クライアント・サーバ方式



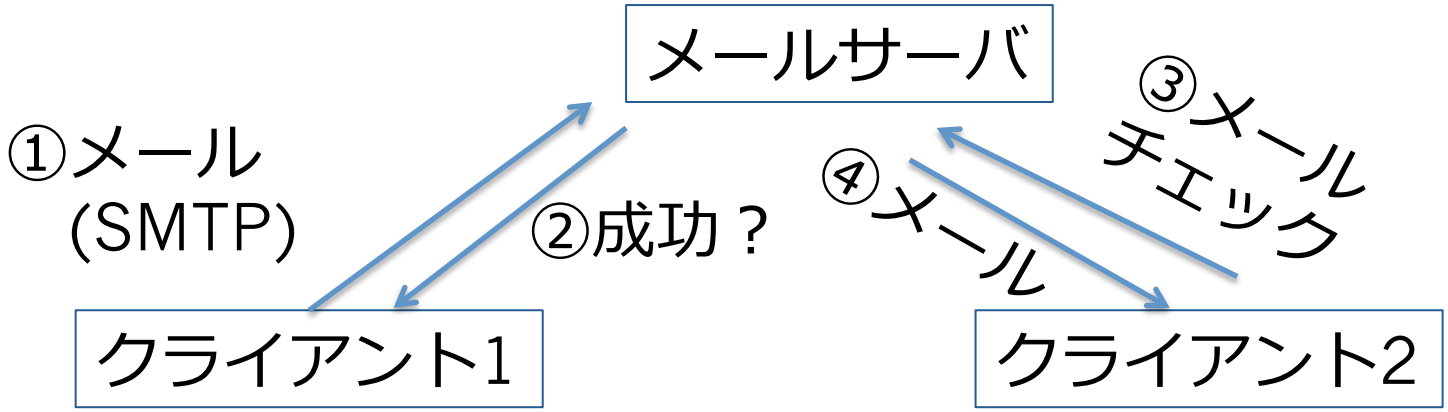
代表的なネットワーク構成の1つ。
データを要求するコンピュータ(=クライアント)と
データを保持し、クライアントに提供する
コンピュータ(=サーバ)に分ける方式

例1) Webサーバ(HTTP)



Webサーバでは、クライアントがURLをリクエストとして送信すると、サーバ側でURLを解析しHTMLを生成・クライアントに返却

例2) メールサーバ(SMTP/POP)



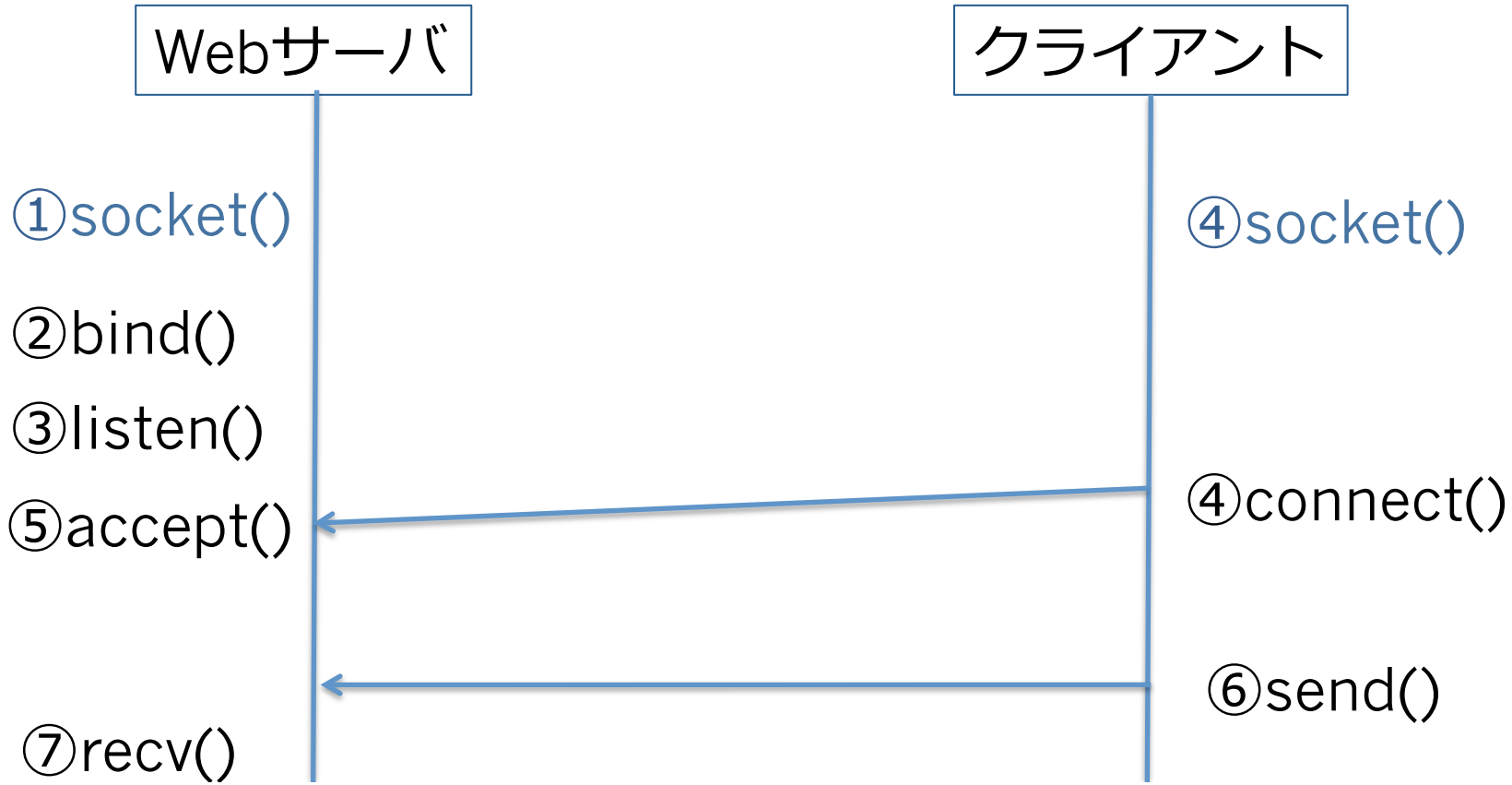
メールサーバでは、クライアントがメール送信のリクエストを行い、メールサーバ側でメールを保持。
送り先のコンピュータがメールチェックし、メールサーバがメールを送信する。

ソケット

- ネットワーク通信の処理をかんたんに記述するための仕組み
  - OSが機能を提供している
  - C言語から呼び出すときにはソケット関数を使う

ソケットプログラミングの流れ

- サーバ構築ではソケット関数を組み合わせプログラムを作成



ソケット関数を使う前準備

- sockaddr_in構造体に必要な情報を保持しておく
  - 主な構成要素
    - sin_family: 通信方式の指定
      (TCPの場合AF_INETという定数を指定)
    - sin_port: ポート番号
    - sin_addr.s_addr: IPアドレス

ソケット関数を使う前準備(例)

コンピュータごとにデータの保持方法（バイトオーダー）が異なり
うまく通信できない。
IPアドレスやポート番号はネットワークバイトオーダーという
共通の形式に変換する

htons(ポート番号): ポート番号をネットワークバイトオーダー
inet_addr(IPアドレス): IPアドレスをネットワークバイトオーダー

```
// sockaddr_in構造体の宣言
struct sockaddr_in server, client;
// serverの情報を代入
server.sin_family = AF_INET;
server.sin_port = htons(2048);
server.sin_addr.s_addr = inet_addr("0.0.0.0");
```

代表的なソケット関数(1/7)

- socket()関数: ソケットを作成する
  - 第1引数：通信プロトコル(通常はAF_INET)
  - 第2引数：通信の種類（通常はSOCK_STREAM）
  - 第3引数：プロトコル詳細設定(通常は0)

  - 戻り値： ソケットの作成に失敗　-1
    ソケットの作成に成功　整数値
    (戻り値をソケットディスクリプタという)

  - 例

```
// ソケットの作成
server_fd = socket(AF_INET, SOCK_STREAM, 0);
```

代表的なソケット関数(2/7)

- bind()関数: ソケットにIPアドレス等を設定
  - 第1引数：ソケット
  - 第2引数：sockaddr_in構造体へのポインタ
  - 第3引数：sockaddr_in構造体のサイズ

  - 戻り値： ソケットの作成に失敗　-1
    ソケットの作成に成功　0

  - 例

```
// sockaddr_in構造体の情報をソケットに関連付け
bind(server_fd, (struct sockaddr *)&server, sizeof(server));
```

代表的なソケット関数(3/7)

- listen()関数: クライアントからの接続待ちを準備
  - 第1引数：ソケット
  - 第2引数：同時接続できるクライアント数

  - 戻り値： 失敗　-1
    成功　0

  - 例

```
// クライアントからの接続待ちを準備
listen(server_fd, 5);
```

代表的なソケット関数(4/7)

- accept()関数: クライアントからの接続を待ち、承認する
  - 第1引数：ソケット
  - 第2引数：クライアントのsockaddr_in構造体へのポインタ
  - 第3引数：構造体のサイズを格納した変数へのポインタ
  - 戻り値： 失敗　-1、成功 整数値

```
// クライアントからの接続待ちを開始
int client_length = sizeof(client);
int client_fd;
client_fd = accept(server_fd,
  (struct sockaddr *)&client, &client_length);
```

代表的なソケット関数(5/7)

- connect()関数: クライアントからサーバへ接続する
  - 第1引数：ソケット
  - 第2引数：サーバのsockaddr_in構造体へのポインタ
  - 第3引数：構造体のサイズを格納した変数へのポインタ
  - 戻り値：　失敗　-1
    成功　整数値

```
connect(socket_fd, (struct sockaddr *)&server,
                                    sizeof(server));
```

代表的なソケット関数(6-7/7)

- send関数：パケットを送信する
- recv関数：パケットを受け取る
- send関数/recv関数の引数と戻り値
  - 第1引数：送り先のソケット
  - 第2引数：文字列配列
  - 第3引数：文字列の長さ
  - 第4引数：動作の設定(通常は0)
  - 戻り値：失敗の場合-1、成功の場合バイト数

代表的なソケット関数(6-7/7)

- send()関数の記述例

```
// メッセージをキーボードから読み込み送信
char string[256]; scanf("%s", string);
send(socket_fd, string, strlen(string)+1, 0);
```

- recv()関数の記述例

```
// クライアントから受け取った文字列を表示
char read_buffer[1000];
recv(client_fd, read_buffer, sizeof(read_buffer)+1, 0);
printf("accept> %s\n", read_buffer);
```

演習概要

- C言語ソケットプログラミング演習
  - プログラムのひな形を事前にダウンロード
    $ cp ~ishigaki/network/server.c server.c
    $ cp ~ishigaki/network/client.c client.c
  - 演習1から演習11：
    簡単なソケット通信の仕組みを作る
    (基本的には手順通り打ち込めばok)
  - 課題12：1人では出来ないので、近くの人に声を掛けて一緒に演習してください。
  - 課題13−14はこれまでのC言語の知識で工夫してみましょう
  - 本演習の解答は講義後にGitHubにて配布

コンパイルと実行方法

- コンパイルの方法
  - サーバ(server.c)
    $ cc server.c –o server
  - クライアント(client.c)
    $ cc client.c –o client
- サーバ/クライアントの起動方法
  - 端末のウィンドウを2つ作成
  - 1つのウィンドウでサーバプログラムを実行
    $ ./server
  - もう一つのウィンドウでクライアントプログラムを実行
    $ ./client

演習1

1. 以下の5つのライブラリを*include*せよ

```
1 #include <stdio.h>
2 #include <sys/types.h>
3 #include <sys/socket.h>
4 #include <netinet/in.h>
5 #include <arpa/inet.h>
6 #include <string.h>
7 #include <unistd.h>
```

演習2

- まずはサーバ側から作成する。
- server.cに以下のようにsockaddr_in構造体を宣言せよ
  (0.0.0.0は自身のIPアドレス)

```
// sockaddr_in構造体の宣言
struct sockaddr_in server, client;
// serverの情報を代入
server.sin_family = AF_INET;
server.sin_port = htons(2048);
server.sin_addr.s_addr = inet_addr("0.0.0.0");
```

演習3

- server.cに以下の処理を記述せよ
  - 以下のようにsocket()関数を用いてソケットを作成しなさい
  - printf()関数でserver_fdの値を表示し正しくソケットが作成されたことを確認せよ（正しい場合はbの値が0になる）

```
// ソケットの作成
int server_fd = socket(AF_INET, SOCK_STREAM, 0);
```

演習4

- server.cに以下の処理を記述せよ
  - bind()関数を用いて以下のようにソケットに対し、接続先を指定しなさい
  - printf()関数でbの値を表示し正しくソケットとIPアドレス、ポート等が関連付けられたことを確認せよ
    （正しくない場合は−1、正しい場合は整数値になる）

```
// sockaddr_in構造体の情報をソケットに関連付け
int b = bind(server_fd, (struct sockaddr *)&server, sizeof(server));
```

演習5

- server.cに以下の処理を記述せよ
  - listen()関数/accept()関数を用いて、クライアントからの接続待ちをできるようにしなさい

```
// クライアントからの接続待ちを準備
listen(server_fd, 5);
// クライアントからの接続待ちを開始
int client_length = sizeof(client);
int client_fd;
client_fd = accept(server_fd,
  (struct sockaddr *)&client, &client_length);
```

演習6

- 演習3まででエラーが出ないことを確認したら、次にクライアントからサーバーへの接続を実装していきます
- client.cに以下の処理を記述しなさい
  - 以下のようにsockaddr_in構造体を宣言しなさい

```
// サーバー情報の宣言
struct sockaddr_in server;
server.sin_family = AF_INET;
server.sin_port = htons(2048);
server.sin_addr.s_addr = inet_addr("0.0.0.0");
```

演習7

- client.cに以下の記述を追加しなさい
  - 以下のようにソケットを作成し、connect()関数でサーバに接続する

```
// ソケット作成と接続
int socket_fd = socket(AF_INET, SOCK_STREAM, 0);
connect(socket_fd, (struct sockaddr *)&server,
            sizeof(server));
```

演習8

- 続いてサーバ側に、クライアントからの接続をaccept(受け入れ)する処理を記述します
- server.cに以下の処理を記述しなさい
  - 以下のようにaccept()関数を記述し、クライアントからの接続が正しくacceptされることを確かめなさい

```
// クライアントからの接続待ちを開始
int client_length = sizeof(client);
int client_fd;
client_fd = accept(server_fd,
  (struct sockaddr *)&client, &client_length);
```

演習9

- クライアント側にパケットを送信する処理を記述します
- client.cに以下の処理を記述しなさい
  - send()関数を用いてキーボードから読み込んだ文字列を送信する処理を記述しなさい

```
// メッセージをキーボードから読み込み送信
char string[256]; scanf("%s", string);
send(socket_fd, string, strlen(string)+1, 0);
```

演習10

- サーバ側に文字列を受け取る処理を記述します
- クライアントから送られ文字列を受け取り、表示する以下の処理を記述しなさい

```
// クライアントから受け取った文字列を表示
char read_buffer[1000];
recv(client_fd, read_buffer, sizeof(read_buffer)+1, 0);
printf("accept> %s\n", read_buffer);
```

演習11

- ソケットを閉じる以下の処理をサーバ側、クライアント側に記述しなさい
  - サーバ側

```
close(server_fd);
```

  - クライアント側

```
close(socket_fd);
```

演習12

- sockaddr_in構造体のIPアドレスを書き換え、サーバとして別のマシンのIPアドレスを指定できるようにしなさい。
  ※ ifconfigコマンドで自身のマシンのIPアドレスを調べることが出来ます。

- 近くの人に声をかけ、異なるマシン間で通信できることを確かめなさい

演習13

- これまでに作成したサーバ/クライアントでは、一度メッセージを送っただけで終了してしまいます。
- while文を用いてクライアントから何度もメッセージを送る事ができるように書き換えなさい

演習14

- ここまでのプログラムではクライアントからサーバーへメッセージを一度しか送ることが出来ません。
- while文を追加し何度も送ることのできるように変更しなさい

演習15

- これまで作成したクライアント/サーバでは、クライアントからデータを送ることは出来ますが、サーバからクライアントにデータを送ることは出来ません。
- クライアントからデータを受け取ったらサーバ側から「received」という文字列を返すようにプログラムを書き換えなさい

次回

- ソフトウェア開発技法
  - ここ数十年でソフトウェアは大規模化
    → 一人で作成することはできないケースが多い
  - 複数人で効率的にソフトウェアを開発するための考え方/技術を学習します