

ソフトウェア設計及び実験

(実習2) ネットワークプログラミング

2019年10月7日

谷岡 広樹

ネットワークプログラミングと私

Network Programming and My Experience

- 谷岡 広樹（たにおか ひろき）

Hiroki Tanioka (Ph.D)

- 情報センター / 知能情報A6グループ・助教

- 得意分野：情報検索、機械学習、自然言語処理など

Information Retrieval, Machine Learning, Natural Language Processing, etc.

- 現在は、情報センター内のセキュリティや認証の研究をする傍ら、
I am struggling security and authentication issues,
 - とくしま動物園で使うカメラアプリを作ったり、
producing a camera app for the Tokushima zoo,
 - 野球やサッカーのデータを分析したり、
analyzing big data of soccer and baseball,
 - 企業と、対話エンジンの精度向上のための共同研究をしています。
and collaborative research about dialogue system with a company.

- 居室：情報センター棟 502号室（ここにはあまりいない。Catch me if you can.）

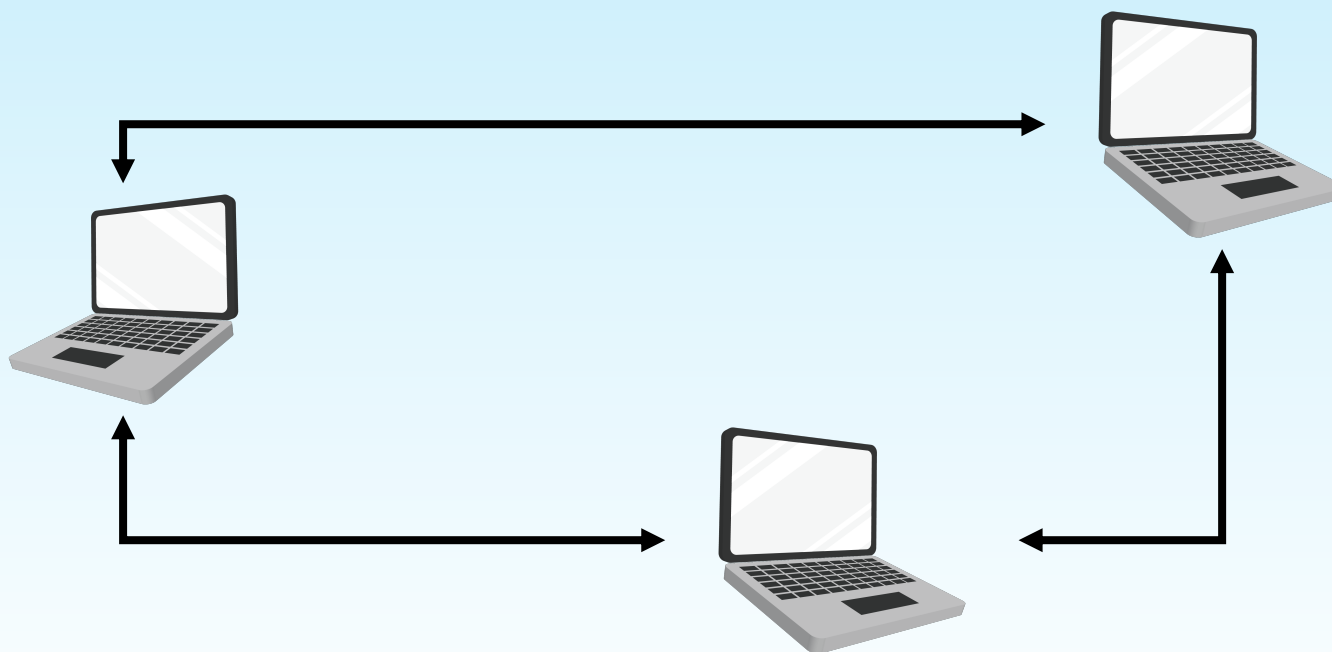
- tanioka.hiroki@tokushima-u.ac.jp
 - 苦情、要望、ファンレターは、Eメールでください。
Please e-mail me.



後期の課題：ネットワーク通信ゲーム！

The 2nd Semester: Network Communication Game!






- 複数プレイヤーがネットワーク通信を使って遊べるゲームを作ります。
Let's make a network communication game played with multi-players.





本日の目的

Today's Mission

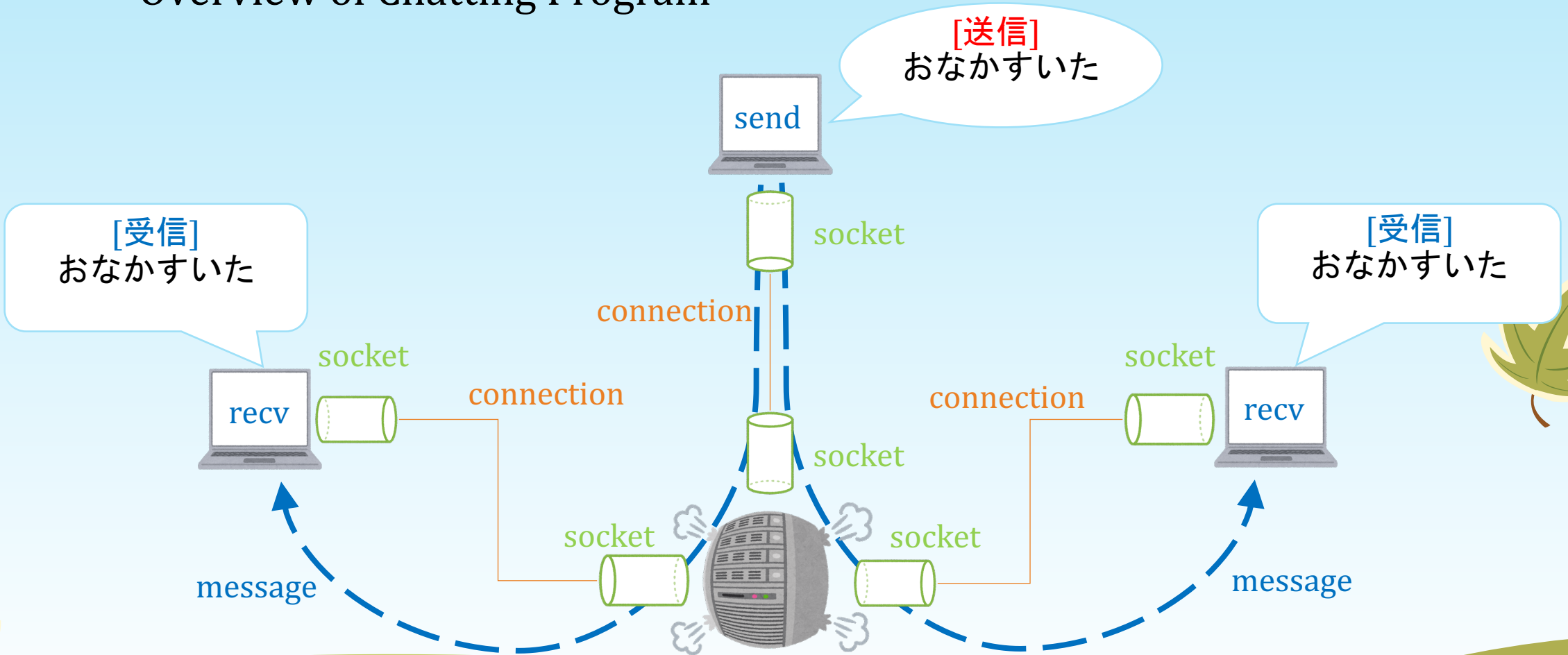
- サンプルプログラムを実行します。
Run a sample program.
 - ソケット通信を用いたネットワークプログラミングについて解説します。
Explain network programming with socket communication.
 - Linux 環境 (on Linux)
 - C言語 (C programming language)
 - ストリームソケット - TCP (Stream Socket - TCP)
 - 演習課題の説明
Explain some exercises.
 - サンプルのチャットプログラムにコメントを入れる。
Please insert comments onto a sample chatting program.
 - そのプログラムを改造する。
Please modify the chatting program.
- 
- 
- 
- 
- 

サンプルプログラムの実行

Run A Sample Program

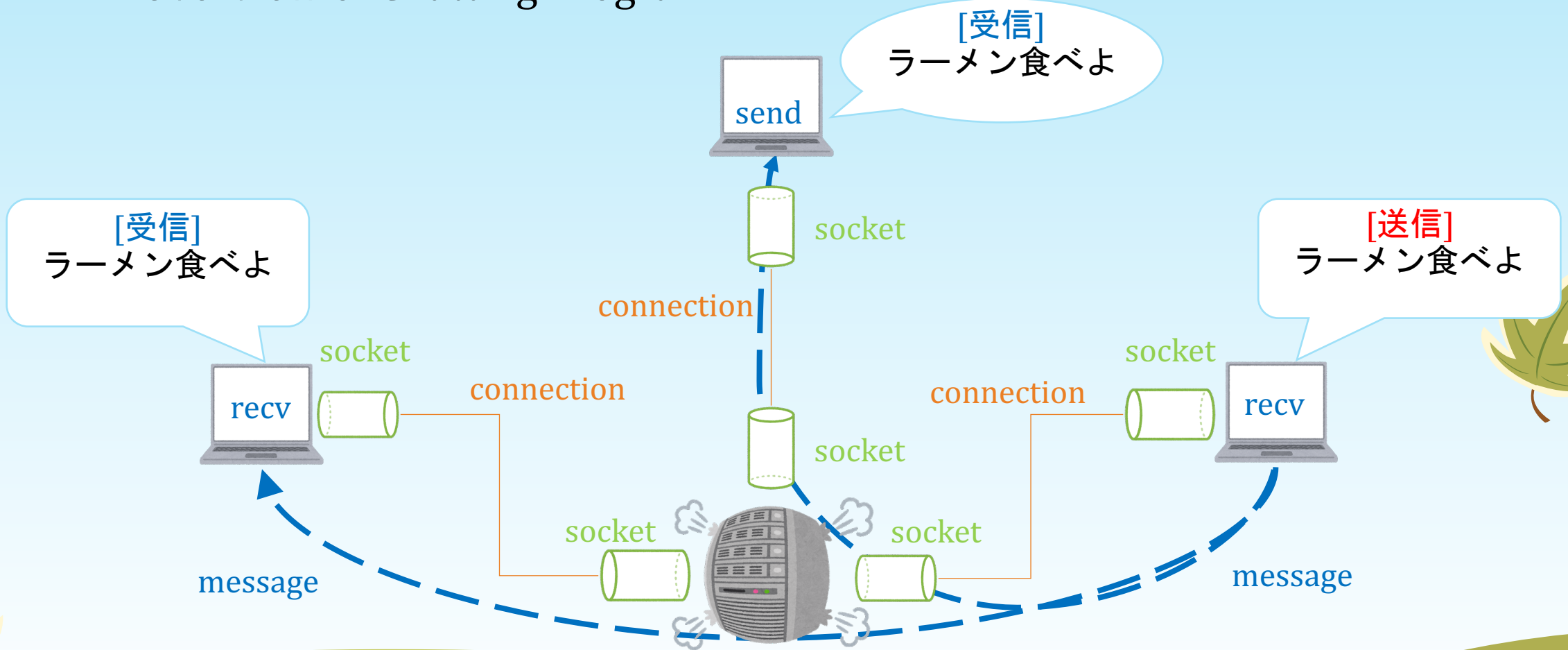
チャットプログラムの概要 (1/2)

Overview of Chatting Program



チャットプログラムの概要 (2/2)

Overview of Chatting Program





サンプルプログラムのコンパイル

Compile the Sample Program

- サンプルプログラムの所在
Where is the sample program?


Manabaコンテンツ(ネットワークプログラミング)

- 次のコマンドで解凍しましょう。
Type the following command for unpackaging.

```
> tar xvfz chat.tar.gz
```

- サーバとクライアントをコンパイルしましょう。
Let's compile a server and a client.

```
> mkdir  
> cd [network]/chat/program1/server  
> make  
> cd ../client  
> make
```



サンプルプログラムの実行

Launch A Server and Two Clients

- 端末1でサーバを起動

Launch a server on a terminal 1.

```
> ./server 2 51xxx
```

- 第1引数: クライアント数 (the number of client)
- 第2引数: ポート番号 (xxx = 学籍番号下3桁) (port number)

- 端末2と端末3でクライアントを起動

Launch two client on a terminal 2 and terminal 3.

```
> ./client [hostname] 51xxx
```

- 第1引数: サーバのホスト名 (clpcxxx), 自身端末の場合は localhost
- 第2引数: ポート番号 (サーバ側で指定したポート番号と同じ)



チャットプログラムの説明

How To Chat ?

- 名前を入力

Launch a server on a terminal 1.

```
> Input your name: Hiroki
```

- コマンドを入力

Input a command.

```
> Input command (M=message, Q=quit):
```





'M' : Send a message.

'Q' : Quit the program.

- メッセージを入力

Send a message.

```
> おなかすいたyo
```



ネットワークプログラミング

Network Programming

ソケットとは？

What's Socket?

- ソケットとは、コンピュータ内で仮想的に実現したネットワーク回線のインターフェースのこと

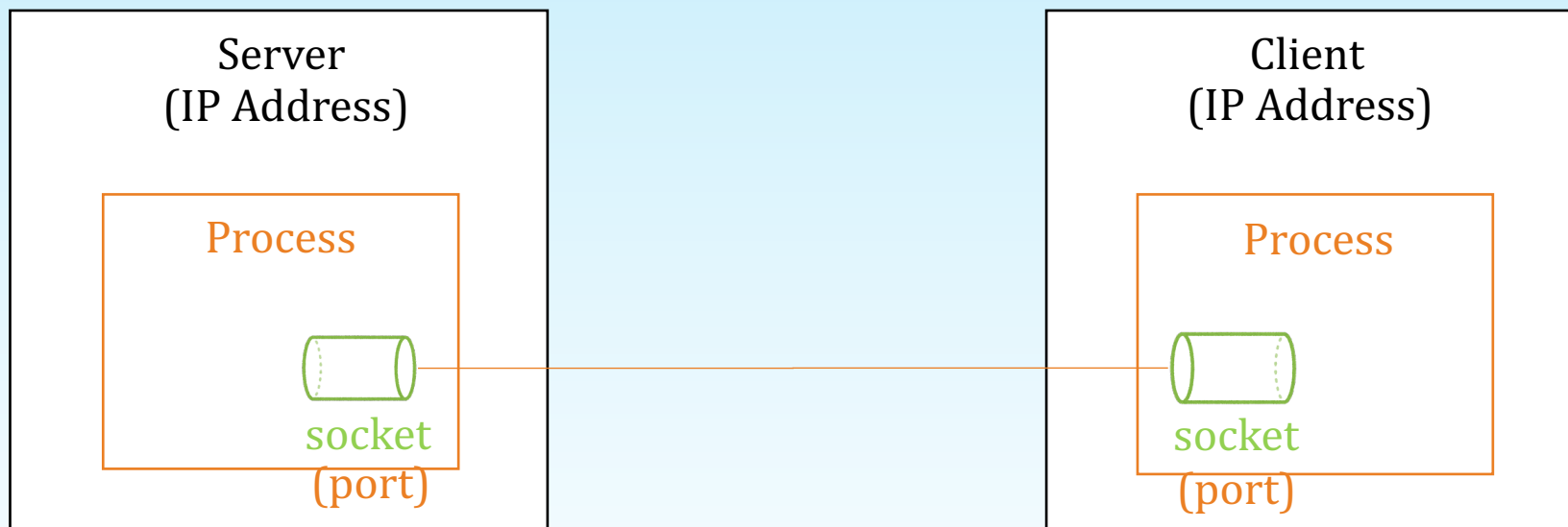
Socket is an interface of virtual network communication realized on a computer.



ソケット通信の概要

Overview of Socket Communication

- ソケット = 送信者と受信者の橋渡しをするデータの出入口
Socket = A bridge interface of data between sender and receiver.





IPアドレスとポート番号

IP Address and Port Number

- IPアドレスは場所を表す

IP Address describes the place where the machine.

```
> Ifconfig
```

```
en0: flags=8863
```

```
inet 192.168.11.150 netmask 0xffffffff broadcast 192.168.11.255
```


```
media: autoselect
```

```
status: active
```

- ポート番号は受付窓口(Port = 港)

Port number is a reception counter of the machine.







- Webブラウザ(HTTP)は 80 番
- メール(POP3)は 110 番



ポート番号

Port Number

1-1024	Well-Known : メジャーなサービスが利用 (Used by measure services)
1025-49151	Registered : Internet Assigned Numbers Authority (IANA) に登録済み
49152-65535	自由に使える (Free to use by any application if you want)





ホスト名とIPアドレス

Host Name and IP Address

- IPアドレスは数字の羅列
IP address consists of numbers.
- ホスト名は文字列の名前
Host name which is the name consists of characters.
clpc001 -> 192.168.1.101
- プログラムではホスト名をIPアドレスに変換する必要があります。
Needed to change a host name to IP address in case of on a program.
> nslookup [host name]

gethostbyname

- ホスト名(clpc001)からIPアドレスを求める(192.168.1.101)
 - ホスト名は
 - /etc/hosts (ローカルに保存されているリスト)から取得
 - DNS(Domain Name System)サービスを利用してインターネットから取得







```
struct hostent *sv_host;  
sv_host = gethostbyname("clpc001");
```

- **引数** = ホスト名
- **戻り値** = ホストに関する情報を持つhostent型構造体へのポインタ
 - sv_host->h_name がホスト名(char *型)



プロトコル(TCPとUDP)

Protocols (TCP and UDP)

- データ通信のためのプロトコルは主に**TCP**と**UDP**の二つ
 - **TCP (Transmission Control Protocol)**
 - 信頼性が高い(データ送受信が正しくできたか逐次チェック)
 - 通信コストが高い
 - ウェブページ, メール, ファイル送受信などに利用
 - **UDP (User Datagram Protocol)**
 - 信頼性が低い(データの欠損などを確認しない)
 - 通信コストが低い
 - 音声・映像配信などの利用
 - 今回は**TCP**のための**ストリームソケット**を使用
- 
- 
- 
- 
- 
- 

ソケット通信の流れ(TCP) – socket()

Flow of Socket Communication (TCP)

クライアント

ソケット生成

socket()

サーバ

ソケット生成

socket()

APIの説明 – socket()

Explaining of API

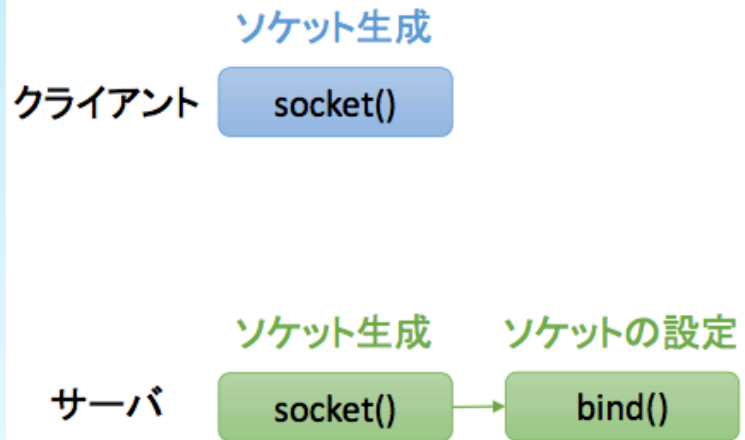
- ストリームソケット

```
sock = socket(AF_INET, SOCK_STREAM, 0); //ソケット生成
```

- 引数1 = ネットワークアドレスの種類(AF_INET=インターネット)
- 引数2 = ソケットの種類(SOCK_STREAM=ストリームソケット(TCP用))
- 引数3 = プロトコル(0=自動選択. TCPが選ばれる.)
- 戻り値 = ソケットトークン(ソケットの識別番号)

ソケット通信の流れ(TCP) – bind()

Flow of Socket Communication (TCP)



APIの説明 – bind()

Explaining of API

- ソケットの基本設定

- socketaddr_in 型構造体を利用

```
socketaddr_in sv_addr;  
memset(&sv_addr, 0, sizeof(sv_addr));  
sv_addr.sin_family = AF_INET;  
sv_addr.sin_port = htons(port);  
sv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
```

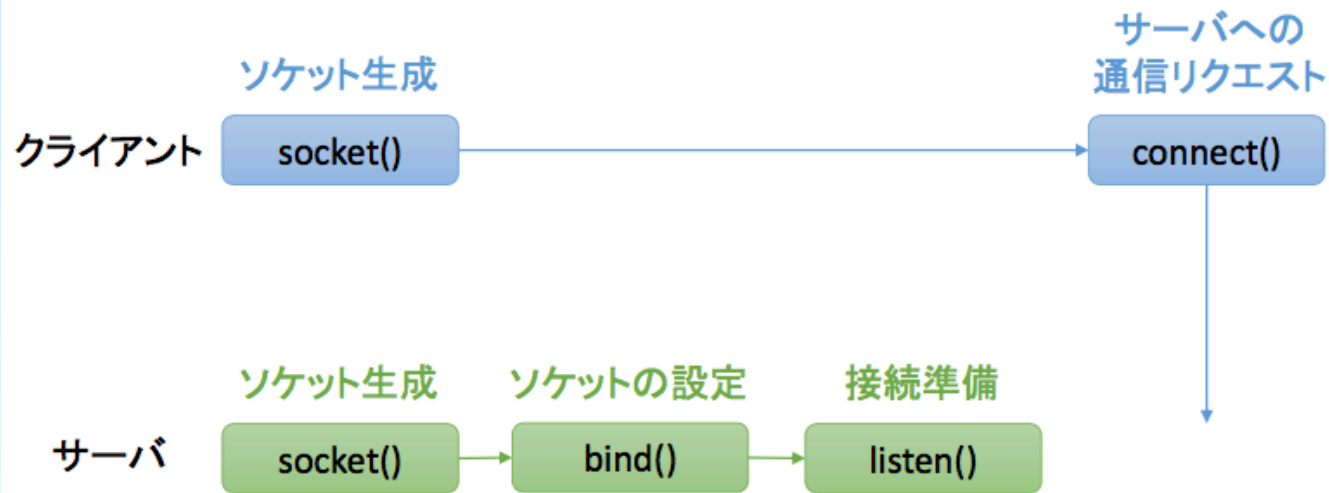
//sv_addrの初期化
//アドレスの種類=インターネット
//ポート番号=port
//INADDR_ANY=任意アドレスから受付可

- ソケットに設定を結びつける

```
bind(soc, &sv_addr, sizeof(sv_addr));
```

ソケット通信の流れ(TCP) – listen(), connect()

Flow of Socket Communication (TCP)



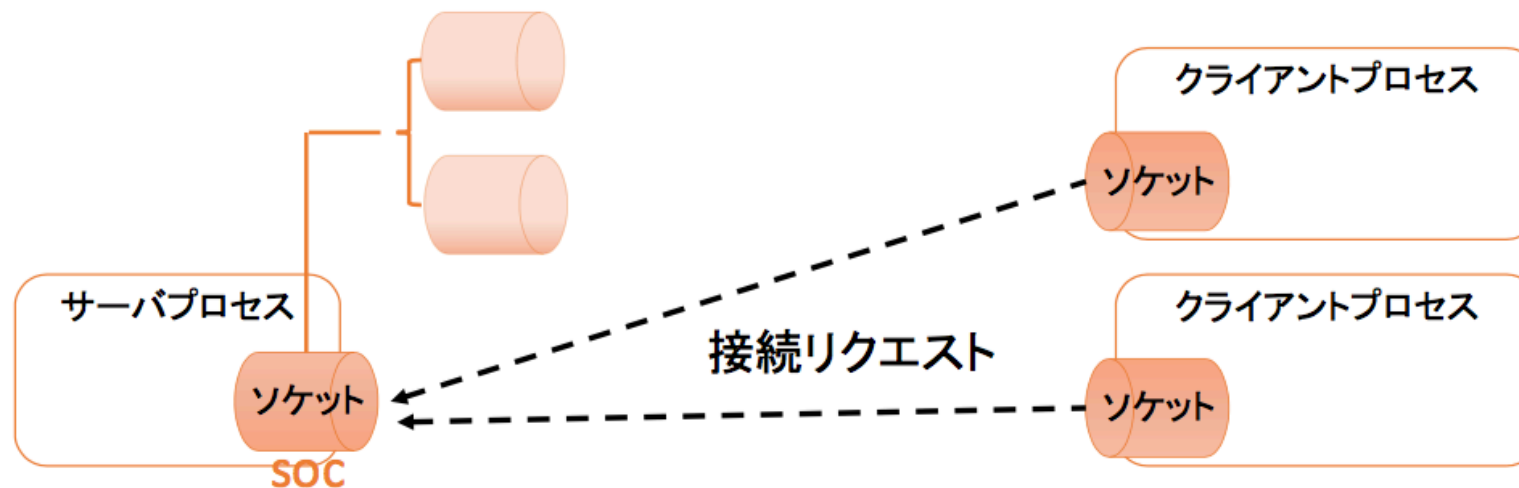
APIの説明 – listen()

Explaining of API

- 接続ソケットの準備

`listen(soc, backlog);`

- 引数1 = 接続要求受付用ソケット
- 引数2 = 接続要求受付最大数



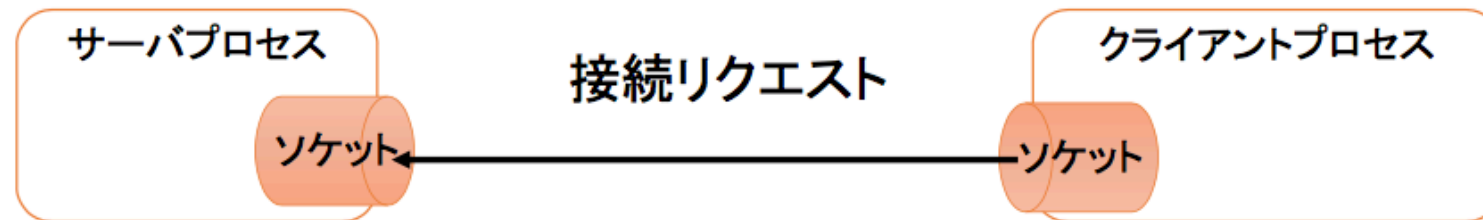
APIの説明 – connect()

Explaining of API

- クライアントからサーバへの接続要求

```
connect(soc, &sv_addr, sizeof(sv_addr));
```

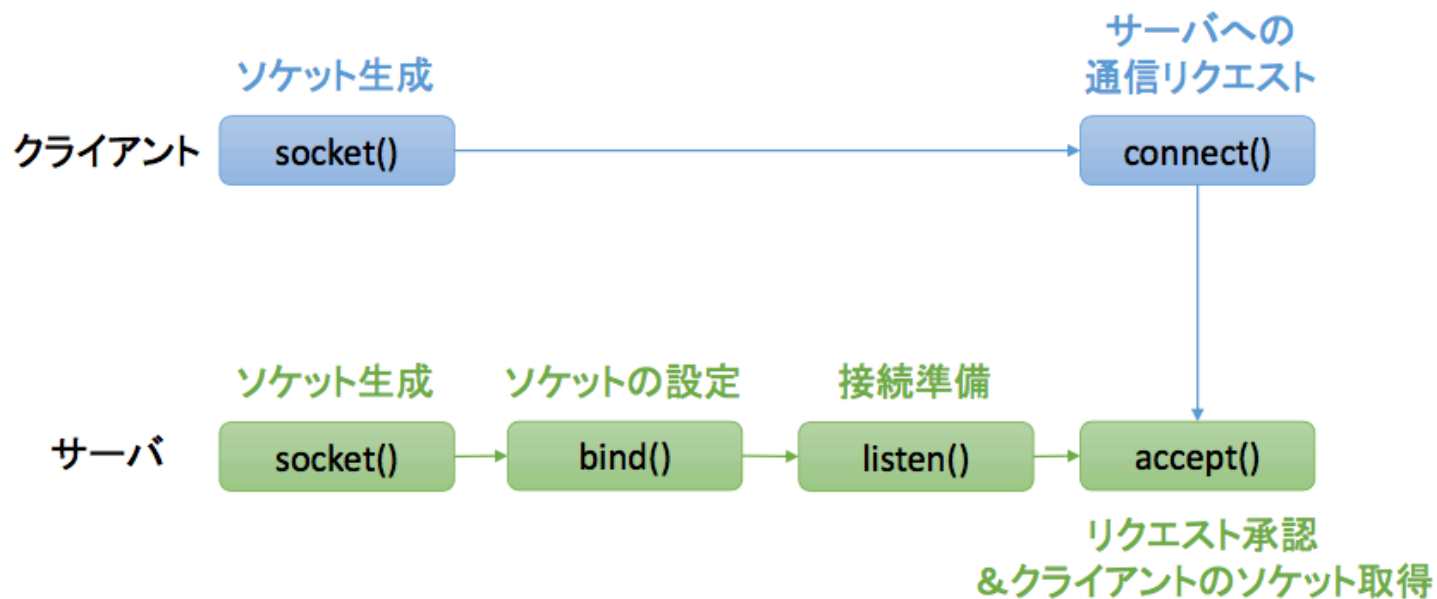
- 引数1 = 接続したいソケットのトークン
- 引数2 = サーバのアドレス情報(socketaddr構造体)
- 引数3 = 引数2のサイズ



- 接続処理が終わるまでクライアントは停止
 - サーバに要求が承諾(accept)されると返り値 = 0

ソケット通信の流れ(TCP) – accept()

Flow of Socket Communication (TCP)



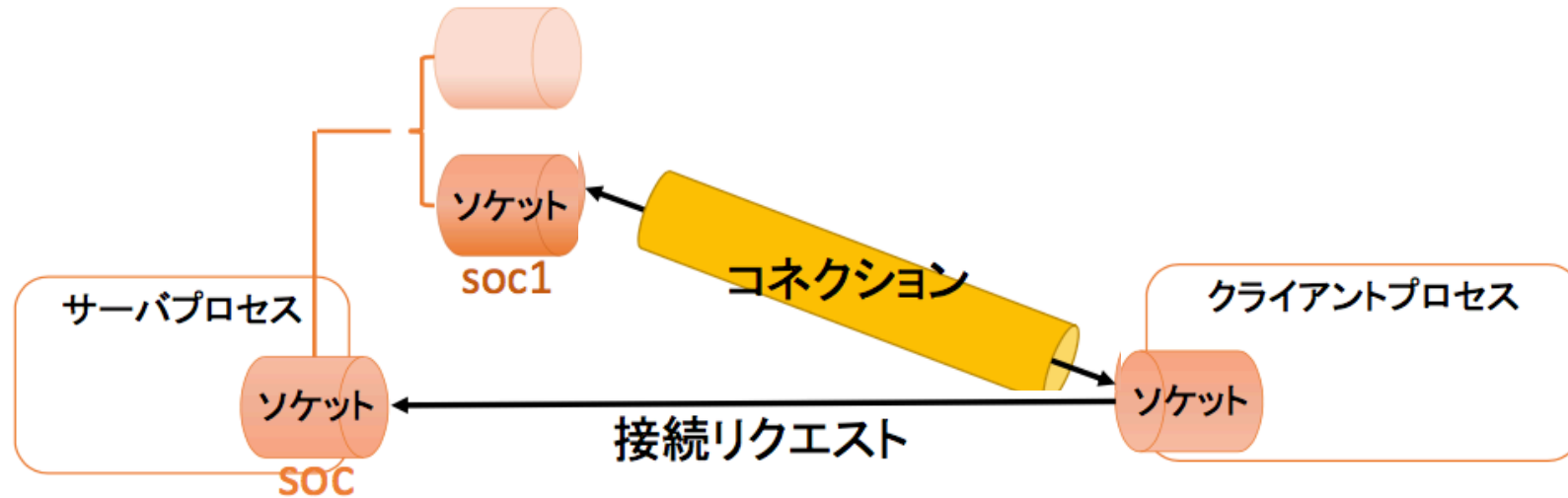
APIの説明 – accept()

Explaining of API

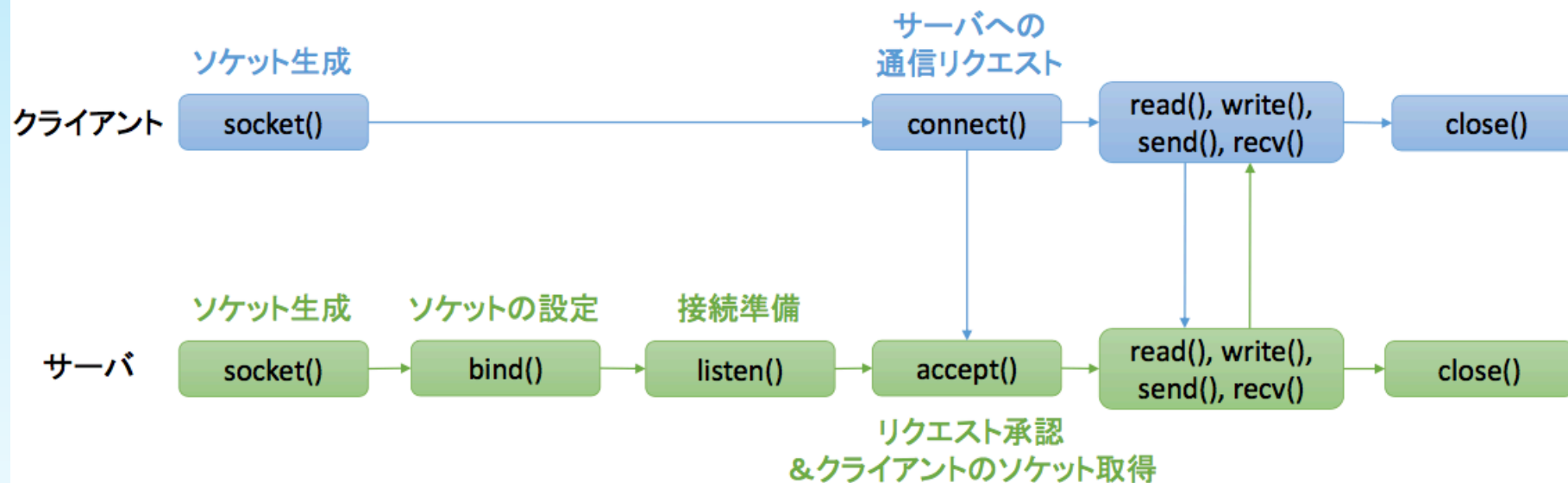
- 接続ソケットの生成と接続の確立

```
soc1 = accept(soc, NULL, NULL);
```

- 引数1 = 接続要求受付用ソケット
- 引数2,3 = 接続先アドレス情報とそのサイズ(NULL=指定無し)



ソケット通信の流れ(TCP) – send(), recv(), read(), write(), select() Flow of Socket Communication (TCP)



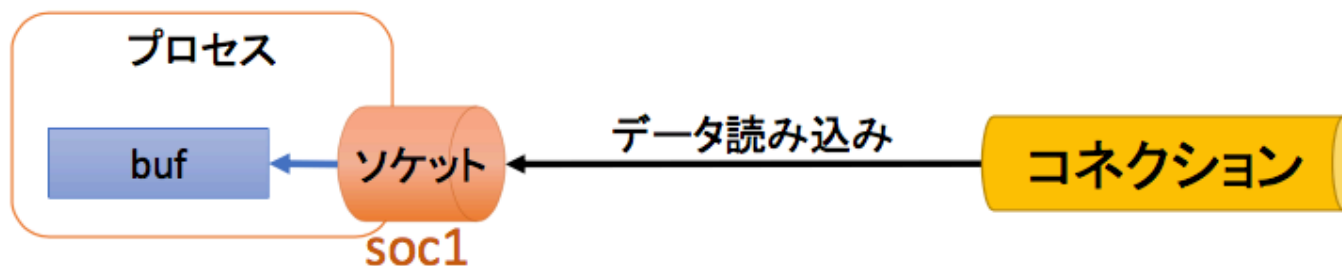
APIの説明 – read()

Explaining of API

- コネクションからバッファへのデータの読み込み

`read(soc1, buf, nbytes)`

- 引数1 = 接続ソケット
- 引数2 = 読み込み先のバッファ
- 引数3 = 読み込みバイト数



ソケット通信の多重化(TCP) – select()

Multiplexing of Socket Communication (TCP)

- ファイルディスクリプタ(以下FD)の集合から読み込み可/書き込み可/例外発生 of FDを発見できる.

select (nfd, readfds, writefds, exceptfds, timeout)

- 引数1 = 最大のFD値+1
- 引数2 = 読み込み可能なFDの集合を表す変数
- 引数3 = 書き込み可能なFDの集合を表す変数
- 引数4 = 例外状態のFDの集合を表す変数
- 引数5 = selectがブロックする最大待ち時間

ファイルディスクリプタ (fd_set)

File Descriptor (fd_set)

fd_set型構造体

- FD集合を表す構造体
 - selectの引数2,3,4の型はfd_setのポインタ

readfdsの内容

0	1	2	3	4	...
0	0	0	0	1	...

FD 0 = 標準入力
FD 1 = 標準出力
FD 2 = 標準エラー出力

select(nfd, &readfds, NULL, NULL, NULL)

readfdsの内容

0	1	2	3	4	...
0	0	0	0	0	...



ファイルディスクリプタを操作するマクロ

Macro Functions for Operating File Descriptor



- FD_ZERO, FD_SET, FD_CLR, FD_ISSET によって操作

FD_ZERO(fds) = fds をゼロクリア

FD_SET(i, fds) = i番目のFDに対応する値を1にセット

FD_CLR(i, fds) = i番目のFDに対応する値を0にクリア

FD_ISSET(i, fds) = i番目のFDが1かどうか確認



The slide features a light blue background with stylized autumn leaves in green and orange scattered around the edges. At the bottom, there are rolling green hills. The main title is centered in a large, bold, black Japanese font, with its English translation below it in a smaller, black, serif font.

プログラミング演習





Programming Exercise



演習課題 (1/4)

Exercises









1. サンプルprogram1のソースに詳細なコメントを付与してください。
 2. サンプルプログラムprogram2のソースを改変し、クライアント間で発言権が順番に移動するチャットシステムを作成してください。
 - クライアント 1 が発言→クライアント 2 が発言→クライアント 3 が発言
→ . . .
 - クライアントの画面には現在の発言者を表示させてください。
 - 発言権のないクライアントが発言した場合にはサーバから発言したクライアントに警告文を送信してください。
- 
- 
- 
- 



演習課題 (2/4)

Exercises

- 
- 
- 
- 
- 
- 
3. [オプション] その他の機能を追加して, program1, program2フォルダと同じ場所にprogram3フォルダを作成してください。
 - 機能追加したチャットシステムのファイルを置くこと
 - 実行時に機能が分かるように使用方法を画面に表示すること
 4. 以下の内容を含むレポートをPDFファイルで作成してください。
 - サンプルに追加した仕様の解説, プログラムの使用方法
 - 実装上の工夫点
 - 講義 & 課題に関する感想

演習課題 (3/4)

Exercises

- 締切 : 2019年10月15日(火) 12:00 JST

Due date : 10,Oct.2017 12:00 JST

- 提出方法 : フォルダ chat 以下にソースコード及びレポート (report.pdf) を保存、圧縮し、[cアカウント].tar.gz というファイル名で Moodle へ提出してください。

How to submit : Save source codes and report (report.pdf) under a folder “chat”, compress the folder into [c account].tar.gz, and submit the file to Moodle.

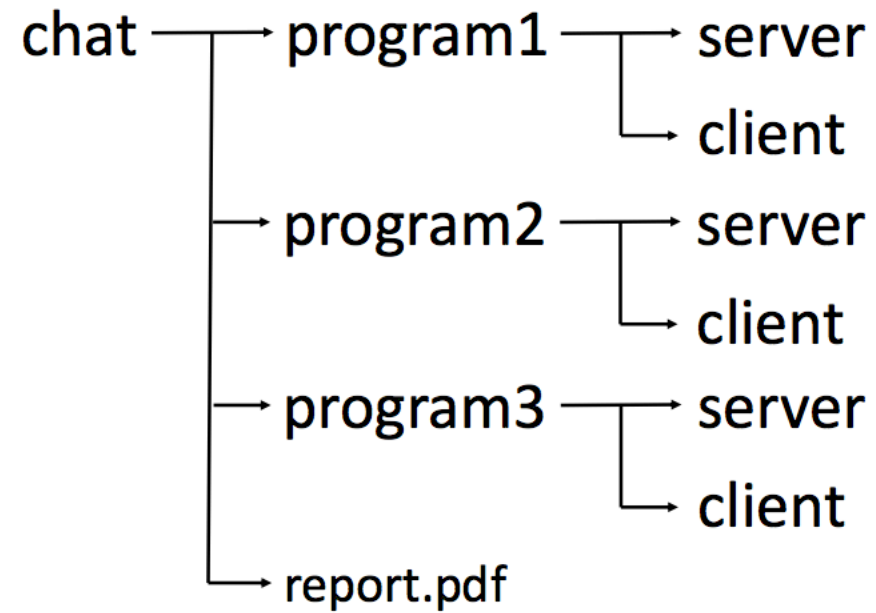
- 注意事項 Attention!!


- 指定されたフォルダ構造
- 半角英数字
- 圧縮形式を確認
- 解凍できることを確認
- 再提出したい場合は、メール連絡すること
- 圧縮ファイルには、実行ファイルやオブジェクトファイル(.o)を含めないこと

演習課題 (4/4)

Exercises

提出ファイルの構造



The slide features a light blue background with stylized autumn leaves in green, orange, and yellow scattered in the corners. At the bottom, there are rolling green hills. The text is centered in the middle of the slide.

では、検討を祈る！
Let's do your best!