研究室レポート５

川瀬 翔大 河村 貴史

1　サーバ-とクライアント間の会話

まず、サーバーが仲介役とならずにクライアントとサーバーでチャットをする。接続時にクライアントに名前を聞き、サーバー側にも名前が表示される。

1.1　サーバー

まず、大まかな流れは

1. 変数の定義
2. ソケットの作成
3. アドレス構造体の設定
4. ソケットと IP アドレスなどを紐づける (bind)
5. 接続待ち状態に (listen)
6. 接続 (accept)
7. 名前受け取りそして表示
8. メッセージのやり取り

である。

１．変数の定義

```
int server_fd, client_fd;
struct sockaddr_in server_addr, client_addr;
socklen_t addr_len = sizeof(client_addr);
char buffer[BUF_SIZE];
char client_name[BUF_SIZE];
```

server_fd、client_fd はソケットの番号（ファイルディスクリプタ）を入れる変数である。sockaddr_in 構造体の server_addr,client_addr は IP アドレスなどを入れる構造体である。addr_len は構造体 client_addr のサイズを表す。あと２つの変数はメッセージ、名前を入れる変数である。

２．ソケットの作成

```
server_fd = socket(AF_INET, SOCK_STREAM, 0);
if (server_fd < 0) {
    perror("socket");
    exit(1);
}
```

関数 socket は以下のような引数をとる

```
socket(アドレスファミリー，ソケットタイプ，特定の通信プロトコル);
```

ここではアドレスファミリーに”AF_INET”を指定する。これは IPv4 を表す。また、ソケットタイプは”SOCK_STREAM”を指定する。これは TCP を使用したい場合に用いる。これはストリーム的にデータをやり取りしたいサービスに使用される。第三引数の特定の通信プロトコルは基本的には「０」でよい。それは、ほとんどの場合、アドレスファミリーとソケットタイプでプロトコルが指定できるためである。

３．アドレス構造体
構造体 server_addr に IP アドレス、ポート番号を指定する。ここでコードの説明をする前に sockaddr_in 構造体を説明する。

```
struct sockaddr_in{
    short sin_family
    unsigned short sin_port
    struct in_addr sin_addr
    char sin_zero
}
```

seever_addr は上記の構造体であるので、１つ１つ指定していく。

```
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(PORT);
server_addr.sin_addr.s_addr = INADDR_ANY;
```

htons は数字（整数）をネットワークで通用する統一された形 (ビックエンディオン) に変換する関数である。sockaddr_in 構造体の中に in_addr 構造体が入っている。in_addr は IPv4 アドレスを格納するシンプルな構造体である。inaddr_any は任意のアドレスを指定している。

４．ソケットと IP アドレスなどを紐づける (bind)

```
if (bind(server_fd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
    perror("bind");
    exit(1);
}
```

bind 関数は作成したソケットと IP アドレス、ポート番号を紐づける関数である。
bind 関数は以下のような引数をとる

bind(作成したソケットのファイルディスクリプタ, ソケットに割り当てる IP アドレスなどへのポインタ, 構造体のサイズ);

第二引数で (struct sockaddr*)server_addr となっているが bind 関数が第二引数の形が決まっているためキャストしている。

５．接続待ち状態にする (listen) listen 関数は作成したソケットを待ち状態にして、クライアントの接続を受け入れるようにする関数である。

```
listen(server_fd,1);
```

第二引数は接続要求の最大待機数である。

６. 接続 (accept)

```
client_fd = accept(server_fd, (struct sockaddr*)&client_addr, &addr_len);
if (client_fd < 0) {
    perror("accept");
    exit(1);
}
```

accept 関数は以下のような引数をとる。

accept(待機ソケット, 受け取る構造体へのポインタ, 構造体のサイズ);

accept 関数は bind 関数を同じように第二引数で構造体 sockaddr のポインタの形を指定している。

７. 名前受け取りここでは名前を受け取り、ようこそ「名前」を返す。

```
read(client_fd, client_name, BUF_SIZE);
printf("クライアント名: %s\n", client_name);
snprintf(buffer, BUF_SIZE, "ようこそ、%s さん!\n", client_name);
write(client_fd, buffer, strlen(buffer));
```

read 関数 write 関数の引数は以下のとおりである。

read(ファイルディスクリプタ、保存する変数、保存する最大サイズ)
write(ファイルディスクリプタ, 書き込む変数, 送信するサイズ)

８. メッセージのやり取り

```
while (1) {
    memset(buffer, 0, BUF_SIZE);
    int bytes = read(client_fd, buffer, BUF_SIZE);
    if (bytes <= 0) break;
    printf("[%s] %s\n", client_name, buffer);
    if (strncmp(buffer, "exit", 4) == 0) {
        printf("%s が退出しました。\n", client_name);
        break;
    }
```

```
        // サーバーからの返信
        printf("[Server] ");
        fflush(stdout);
        fgets(buffer, BUF_SIZE, stdin);
        write(client_fd, buffer, strlen(buffer));
    }
```

まず memset 関数で buffer をゼロクリアします。その後、メッセージを読み込み出力します。ユーザーからの入力を待ち、それを受け取った瞬間にクライアントに転送する。

1.2 クライアント

大まかな流れは、

1. 変数の定義
2. ソケットの作成
3. アドレス構造体の設定
4. 通信経路の確率
5. 名前のやり取り
6. メッセージのやり取り

である。1〜3 まではほぼサーバープログラムを同じである。ここでは４からの説明をする。

４、通信経路の確率
ここではソケットの通信経路の確率をする。

```
    if (connect(sock, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
        perror("connect"); exit(1);
    }
```

connect 関数の引数は以下のとおりである。

connect(接続を開始するソケットディスクリプタ, 接続先のサーバーのアドレス情報, 構造体のサイズ);

第二引数は型が指定されているため、キャストしている。

５，名前のやり取り

```
    printf("あなたの名前を入力してください: ");
    fgets(name, BUF_SIZE, stdin);
    name[strcspn(name, "\n")] = 0; // 改行削除
    // 名前をサーバーに送る
    write(sock, name, strlen(name));
```

```
// サーバーからの歓迎メッセージを受信
read(sock, buffer, BUF_SIZE);
printf("%s", buffer);
```

ここでは、書き込まれた名前を読み込み、名前をサーバーに送る。その後、サーバーからの歓迎メッセージを受信し、出力する。

6，メッセージのやり取り
ここでは、メッセージのやり取りをする。

```
while (1) {
    printf("%s: ", name);
    fgets(buffer, BUF_SIZE, stdin);
    write(sock, buffer, strlen(buffer));
    if (strncmp(buffer, "exit", 4) == 0) break;
    memset(buffer, 0, BUF_SIZE);
    read(sock, buffer, BUF_SIZE);
    printf("Server: %s", buffer);
}
```

メッセージを読み込み、サーバーに送信、その後、サーバーからメッセージの受け取るためにバッファをゼロクリアします。そのあと、サーバーからのメッセージを受け取ります。最後に”Server”というプレフィックスをつけて画面に出力する。

2　課題

リアルタイム制にしサーバーが仲介役となりクライアントとクライアントのチャットができるようにする。

References

[1] だえうホームページ,https://daeudaeu.com/socket/,2025 年 10 月 22 日

[2] だえうホームページ,https://daeudaeu.com/fflush/,2025 年 10 月 22 日

[3] CEEKBLOCKS,https://af-e.net/c-language-memset/,2025 年 10 月 22 日

[4]