# 情報科学演習 C 課題 2 レポート

担当教員：小島 英春
提出者：小山 亮
学籍番号：09B15028
e-mail：u745409b@ecs.osaka-u.ac.jp

提出年月日：平成 29 年 5 月 29 日

# 1 課題内容

2-1 echoserver プログラムと通信する以下のような echoclient プログラムを作成せよ。echoclient プログラムは host で動作している echoserver プログラムとポート番号 10120 を介して接続し、標準入力から読み込んだ文字列を echoserver プログラムへ送信し、受信した文字列を標準出力に書き出す。これを標準入力から EOF を受け取るまで (標準入力が端末なら Ctrl-D を入力するまで) 繰り返す。

2-2 simple-talk-server プログラムを完成させよ。また、サーバ側と同様に select() システムコールを用いて、クライアント側のプログラム simple-talk-client を作成せよ。最低限、以下の仕様を満たすプログラムとすること。

- simple-talk-server を起動してから、simple-talk-server が動作しているホスト名を指定して、simple-talk-client を起動することで接続が確立する。
- 接続確立後は、simple-talk-server 側で標準入力から入力した文字列が即座に simple-talk-client 側に表示される。逆も同様。
- 全二重通信で実現すること。すなわち、会話をする 2 人がどのような順序で発言を行っても、相手側に即座にそのメッセージが表示されること。

# 2 TCP 通信プログラム

本課題では TCP 通信を用いたプログラムを作成した。そのプログラムに共通する接続を行う部分について説明する。

## 2.1 TCP クライアントプログラム

TCP クライアントプログラムの接続を確立させるまでの大まかな手順は以下のとおりである。

1. socket() で通信の出入口となるソケットを生成する.。
2. connect() で host と接続する。

socket() について実際に用いたコードを以下に示す。

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

socket() の第一引数には、ドメインの種類である INET を指定している。第二引数には、ソケットの型としてストリーム型を指定している。第三引数には、通信プロトコルとして TCP を指定している。 socket() は与えられた引数に基づいてソケットを生成する。ソケットの生成がうまくいかなかった場合は、エラーを出力し、プログラムを終了するようにしている。次に connect() について実際に用いたコードを以下に示す。

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

connect() の第一引数には、socket() でソケットを生成したときの帰り値を与えている。第二引数には、プログラム実行時に指定されたホスト名から得たアドレスを与えている。第三引数には、接続先のソケットアドレスの大きさを与えている。 connect() は与えられたソケットを与えられたホスト情報のアドレスに接続している。接続がうまくいかなかった場合は、エラーを出力し、プログラムを終了するようにしている。

## 2.2 TCP サーバプログラム

TCP サーバプログラムの接続を確立させるまでの大まかな手順は以下のとおりである。

1. socket() で通信の出入口となるソケットを生成する.。
2. bind() でソケットとポートを対応づける。クライアントはここで対応づけたポートを介してサーバプログラムと接続する。
3. listen() で最大接続待ち数を指定する。
4. accept() でクライアントからの接続要求を待ち、到着したら新たにソケットを生成する。

socket() に関しては第 2.1 節で説明したものと同じである。 bind() について実際に用いたコードを以下に示す。

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

bind() の第一引数には、socket() でソケットを生成したときの帰り値を与えている。第二引数には、ソケットに割り当てるアドレスを与えている。第三引数には、ソケットに割り当てるアドレスのソケットアドレスの大きさを与えている。 bind() は与えられたソケットに、与えられたソケットアドレスを割り当てている。割り当てがうまくいかなかった場合は、エラーを出力し、プログラムを終了するようにしている。listen() について実際に用いたコードを以下に示す。

```
if (listen(sock,5)<0) {
  perror("listen");
  exit(1);
}
```

listen() 第一引数には、socket() でソケットを生成したときの帰り値を与えている。第二引数には、待ち受け数の最大値を与えている。listen() は、待ち受けクライアントの最大値を設定してい

る。待ち受けクライアント数の設定がうまくいかなかった場合は、エラーを出力し、プログラムを終了するようにしている。accept() について実際に用いたコードを以下に示す。

```
if ( ( csock = accept(sock,(struct sockaddr *)&clt,&clen) ) <0 ) {
  perror("accept");
  exit(2);
}
```

accept() の第一引数には、socket() でソケットを生成したときの帰り値を与えている。第二引数には、クライアント側のアドレスを与えている。第三引数には、クライアント側のアドレスのソケットアドレスの大きさを与えている。接続の確立がうまくいかなかった場合は、エラーを出力し、プログラムを終了するようにしている。

# 3 課題 [2-1]

echoserver プログラムと通信する echoclient プログラムを作成した。

## 3.1 仕様

echoclient　プログラムの仕様を記述する。

- echoserver プログラムを実行しているホスト名を第一引数に指定し、実行する。
- 標準入力から文字列を入力し、 echoserver から帰ってきた文字列を標準出力に出力する。
- EOF を入力すると、その時点でプログラムを終了する。
- サーバ側のプログラムが終了するなどして接続が切れた場合は、次の文字を入力した時点で、server closed と出力しプログラムを終了する。
- 入力可能な文字数は '\0' を含め、1024 文字である。

## 3.2 実行結果

```
r-koyama@exp046:~/enshuuC/kadai2$ ./echoclient exp046
aa
aa
bbb
bbb
cc
cc



r-koyama@exp046:~/enshuuC/kadai2$
```

入力ごとに帰ってきた文字列を出力している。改行のみを入力した際もそのまま出力されている。最後に EOF を入力しプログラムが終了している。

# 4 課題 [2-2]

簡易版の talk をサーバ側とクライアント側を分けて作成した。

## 4.1 仕様

simple-talk-server　プログラムと simple-talk-client プログラムの仕様を記述する。

- simple-talk-server　プログラムを実行しているホスト名を第一引数に指定し、simple-talk-client プログラムを実行する。
- 正常に実行されると、両プログラムで connected と出力し、接続したことを伝える。
- 接続されると、双方の動作は同じである。
- 標準入力から入力された文字列を相手側へ送信する。
- 受信側は、文字列を受け取ると、その文字列の頭に相手のホスト名を付けて出力する。
- simple-talk-client プログラムは EOF を入力すると、simple-talk-server　プログラムとの接続を切断し、終了する。
- simple-talk-server　プログラムはクライアントとの接続が切れると、接続待ち状態に戻る。

## 4.2 実行結果

simple-talk-server　プログラムと simple-talk-client プログラムの実行結果を記す。

simple-talk-server　プログラム

```
r-koyama@exp001:~/enshuuC/kadai2$ ./simple-talk-server
connected
exp046 : one
two
exp046 : three
exp046 : four
five
six
exp046 : seven
closed
```

simple-talk-client プログラム

```
r-koyama@exp046:~/enshuuC/kadai2$ ./simple-talk-client exp001
connected
one
exp001 : two
three
four
exp001 : five
exp001 : six
seven

close
r-koyama@exp046:~/enshuuC/kadai2$
```

接続された 2 つのプログラムの実行結果である。まず始めに双方で接続の際に connected と出力されている。クライアント側から文字列を送り始めている。どちらが送信した文字列なのかは、ホスト名が出力されているかどうかで判断できる。一緒にホスト名が出力されている方が受信した文字列である。送信した文字列はその順番を崩すことなく出力されている。最後にクライアント側が EOF を入力し、接続が切れ、simple-talk-client プログラムは終了し、simple-talk-server　プログラムは接続待ち状態に戻っている。

## 5　考察

クライアント側プログラムでホスト名を指定するときに、IP アドレスを用いて実行しても、ホスト名を得ることができたので、gethostbyaddr() は、DNS サーバとの通信を行っているのではないかと考えた。

## 6　感想

他のマシンとの通信を自分のプログラムで実現できたのは、良い経験だった。サーバー、クライアントなどの関係性もわかるようになったという点で、成長を生んだ。