情報科学演習 C レポート 2

藤田 勇樹
大阪大学 基礎工学部 情報科学科 ソフトウェア科学コース
学籍番号: 09B16068
メールアドレス: u461566g@ecs.osaka-u.ac.jp
担当教員
小島 英春 助教授
内山 彰 助教授

提出日: 2018 年 5 月 28 日

1　課題 2-1

1.1　概要

この課題では，引数で与えられたホスト名のホスト上で動作している echoserver と通信し，標準入力から受け取った文字列を echoserver に送信し，echoserver から返された文字列をそのまま標準出力に表示するプログラム echoclient を作成する．echoserver と echoclient の通信には TCP を用いる．

1.2　仕様

このプログラムの動作の流れは以下の通りである．

1. 引数で与えられたホストで動作している echoserver に接続する．
2. 標準入力からの入力を受け付ける．
3. 標準入力の内容が EOF(Ctrl-D) なら echoserver との接続を切りプログラムを終了する．
4. 標準入力の内容を echoserver に送信する．
5. echoserver から返された文字列を表示する．
6. 2. に戻る．

以降の節では，1. の接続と 4. および 5. のデータの送受信について実装内容を説明する．

1.3　接続の確立

1.3.1　ソケットの生成

まず，通信の出入り口であるソケットを生成する．これには `socket()` システムコールを用い，以下のように使用する．

```
sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP)
```

第一引数には通信方法を決定するプロトコルファミリーを指定する．ここでは IPv4 を使用するため `AF_INET` を指定している．第二引数にはソケットの型を指定する．このプログラムでは TCP を使用し，TCP は通常全二重バイトストリームのため `SOCK_STREAM` を指定する．第三引数ではプロトコルを指定するため，`IPPROTO_TCP` を指定している．

なお，この後の `setsockopt()` はソケットを再利用するためのおまじないであり，ここでは解説しない．

1.3.2　接続先の設定

次に，接続先を設定し実際に接続を行う．ここではまず，ソケットの接続先の情報を格納する変数 `svr` に対し，メンバ `sin_family` に `AF_INET`(IPv4 を使用する) を設定し，メンバ `sin_port` に使用するポート番号を設定する．ただし，これに設定する値はネットワーク上で使用するバイトオーダーで記述しなければならないため，`htons()` 関数でバイトオーダーを変更している．最後に，`gethostbyname()` 関数で取得できる接続先ホストの IP アドレスを `sin_addr` メンバに設定する．

最後にここまでに設定した情報をもとに接続を行う．これには `connect()` システムコールを用いる．

1.4 文字列の送受信

接続が完了したら，変数 sock をファイルディスクリプタとして read() および write() システムコールを用いてソケットへの読み書きを行う．

1.5 接続の切断

接続を終了するには close() システムコールを用いる．

1.6 実行結果

キーボードからの入力には頭に>をつけてある．実際の実行時には表示されない．

```
$ ./echoclient exp101
connected with exp101
>hello
hello
>AiUeO
AiUeO
>にほんごニホンゴ日本語
にほんごニホンゴ日本語
>!"#$%&'(),./\
!"#$%&'(),./\
(Ctrl-D を入力)$
```

1.7 発展課題:lowerechoserver

発展課題として，echoserver を改造し，echoclient から送られた文字列のうち大文字を小文字にして返す lowerechoserver を作成した．

具体的には，echoclient から受け取った文字列 rbuf に対し，大文字を小文字に変換する関数 lower を適用してから echoclient に送り返すようにした．

1.7.1 実行結果

```
$ ./echoclient exp101
connected with exp101
>hello
hello
>AiUeO
aiueo
>にほんごニホンゴ日本語
```

```
にほんごニホンゴ日本語
>!"#$%&'(),./\
!"#$%&'(),./\
(Ctrl-D を入力)$
```

2 課題 2-2

2.1 概要

この課題では，一方の端末に入力された文字列を他方の端末に送信して表示するプログラム simple-talk-server と simple-talk-client を作成する．このうち server は `listen()` で接続要求を待つ側，client は `connect()` で接続要求を発する側とする．

2.2 simple-talk-server

2.2.1 接続の確立

simple-talk-client からの接続を待機するまでの部分は，echoserver と同様に，

1. `socket()` でソケットを生成する．
2. 接続先情報を設定する．
3. ソケットにアドレスを `bind()` する．
4. `listen()` で接続待ちの数を指定する．
5. `accept()` でクライアントからの接続を待機する．

の手順で行う．

2.2.2 文字列の送受信

文字列の送受信には echoserver および echoclient と同様にソケットを介して行うが，これらと異なる点は，ソケットと標準入力の両方の入力を同時に受け付けなければならない点である．

これを実現するために `select()` システムコールを用いる．これは，`fd_set` 型の変数に登録したファイルディスクリプタを同時に監視する関数である．ここでは，標準入力である 0 と `socket()` で生成したソケット `sock` をマクロ `FD_SET` で登録し，`select()` でこれらを監視し，標準入力からの入力であればソケットにそれを書き込み，ソケットからの入力であればそれを標準出力に出力する．

2.3 simple-talk-client

simple-talk-server への接続を行う部分は，echoclient と同様に，

1. `socket()` でソケットを生成する．
2. 接続先情報を設定する．
3. `connect()` で接続を行う．

の手順で行う．

また，文字列の送受信には simple-talk-server と同様に `select()` システムコールを用いた．

2.4 発展課題:発言者の名前表示

発展課題として，simple-talk-server と simple-talk-client に，接続時に名前を入力し，発言時に発言者の名前を表示するという拡張を行った．

ここでは，それぞれのプログラムの起動時に名前を入力させ，接続確立時にお互いに名前を送信し，“(名前) > (入力内容)” のように会話の内容を表示するようにした．

なお，標準入力の入力待ちの間は，シェルのプロンプトのように，“(自分の名前) > ” と表示するが，このときにソケットからの入力を感知したら，エスケープシーケンス`\e[1K` で “(自分の名前) > ” を端末の表示から削除し，`\r` でカーソルを行頭に戻してから “(相手の名前) > (相手の入力内容)” と表示するようにした．

2.5 実行結果

simple-talk-server を実行した端末からの出力は以下のようになった．

```
$ ./simple-talk-server
Enter your name>SVR
[exp025.exp.ics.es.osaka-u.ac.jp]
connected with CLT
CLT > hello
CLT > AiUeO
SVR > Hey
SVR > 日本語
CLT > !"#$
closed
```

一方，これと通信した simple-talk-client からの出力は以下である．

```
$ ./simple-talk-client exp101
Enter your name>CLT
connected with SVR
CLT > hello
CLT > AiUeO
SVR > Hey
SVR > 日本語
CLT > !"#$
closed
$
```

3　考察

発展課題として lowerechoserver を作成したが，単純に各バイトが大文字の範囲であれば小文字に変換するという実装のため，文字コードによっては，日本語等のマルチバイト文字に対し特定のバイトを大文字と認識し不本意な変換を行ってしまう可能性があると考えられる．

4　感想

この課題を通して，ネットワークの接続時に実際に動作しているシステムコールを直に知ることができた．