情報工学実験1

「ネットワークプログラミング基礎」

組　：　1組
学籍番号：　０９５７０７Ｂ
氏　　名：　大城佳明

締切日：　2010年11月1日
提出日：　2010年10月29日

1　課題１：サンプルプログラムの実行

サンプルプログラムを自分の実験環境で動作するようにして実行結果を示すとともに、プログラムの中で使われているソケット関連の関数の動作を中心に、クライアントおよびサーバーの動作をフローチャートを用いて解説せよ。

1.1　実行結果

実行結果（server.c）

```
yoshiaki-oshiro-no-macbook-2:network yoshiaki$ ./server 10000
receive 'hogehoge'
send hogehoge
receive 'hello'
send hello
receive 'hello'
send hello
```

実行結果（client.c）

```
yoshiaki-oshiro-no-macbook-2:network yoshiaki$ ./client 127.0.0.1 10000
connected to '127.0.0.1'
TCP> hogehoge
hogehoge
TCP> hello
hello
```

1. server.c を 1000 ポートで実行させる
2. clinet.c を 127.0.0.1(local) の 1000 ポートで実行する
3. clinet.c から文字を入力し、return キーを押す
4. serve.c に入力した文字列が表示される
5. client.c にも同じ文字列が表示される

1.2　説明

1.2.1　server.c

1. 以下の図 1 に server.c のフローチャートを示した。
2. server.c は引数を１つ必要とする。
3. make_socket 関数を使い、ソケットを作成する

(a) make_soket ではポート番号を受け取り、ソケットを作成する

(b) memset() でメモリーの確保を行う

(c) getaddrinfo() で IP での名前解決を行なう

(d) socket() でソケットを作成すると同時に、そのソケットに 識別用の数字を割り振る

(e) bind() はコネクションを受けつける IP アドレス・ポート番号と ソケットとを対応づける

(f) freeaddrinfo() はメモリー解放する

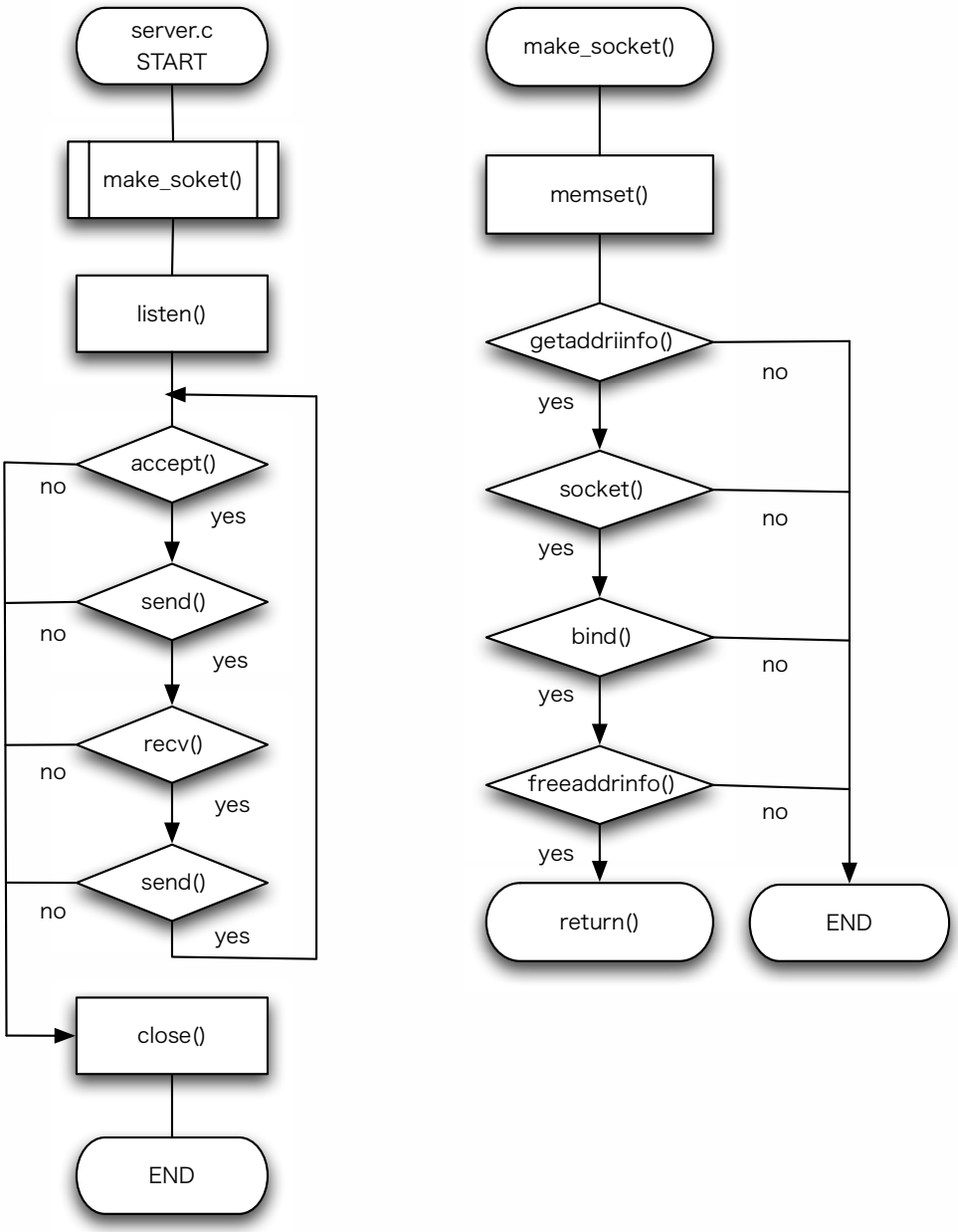

図 1: フローチャート (server.c)

4. listen() で保留中の要求を格納するキューをサーバ側で作る

5. while 文で繰り返し行われる。
   (a) accept() はクライアントからの接続を受け付ける
   (b) send() でクライアント側に送信する
   (c) recv() で TCP で connect、 accept 済みの相手からのパケットを受信する
   (d) 受信した文字列を別の変数に格納して、send() で送り返す
6. 以上のことを繰り返すことにより、クライアントからの送信を server は受信し、クライアントに送信することができる
7. 送受信失敗などで while 文を抜けると、close() で閉じて、エラー文を返す

1.2.2 client.c

1. 以下の図 2 に client.c のフローチャートを示した
2. client.c は引数を 2 つ必要とする
3. connect_server() でソケットを作り、通信接続を行っている
   (a) connect_server() では hostname とポート番号を受け取る
   (b) make_socket を似たような動きをする
   (c) ソケットを作成する
   (d) make_socket とは違いは bind() を使うか、connect() を使うかの違いである
   (e) connect() で生成したソケットをサー バ側プログラムのソケットと通信接続を行う
4. while で繰り返し行われる
   (a) FD_ZERO() は監視対象のディスクリプタ一覧をゼロクリアをする
   (b) FD_SET() はリスニングソケットを監視対象に追加する
   (c) select() で入出力の多重化が可能となる
   (d) FD_ISSET() で入力されたか確認する
   (e) strcmp() は文字列の大小関係の比較を行う。(quit を入力したら終了となる)
   (f) send() でサーバーに送る
   (g) FD_ISSET() で受信を確認する
   (h) recv() でサーバー側からの受信をする
   (i) fflush() でバッファに格納されているデータを吐き出す
5. 以上のことを繰り返すことにより、入力した文字列を送り、受け取ることができる
6. whie 文を抜けると、quit という文字列をサーバーに送り、終了する。

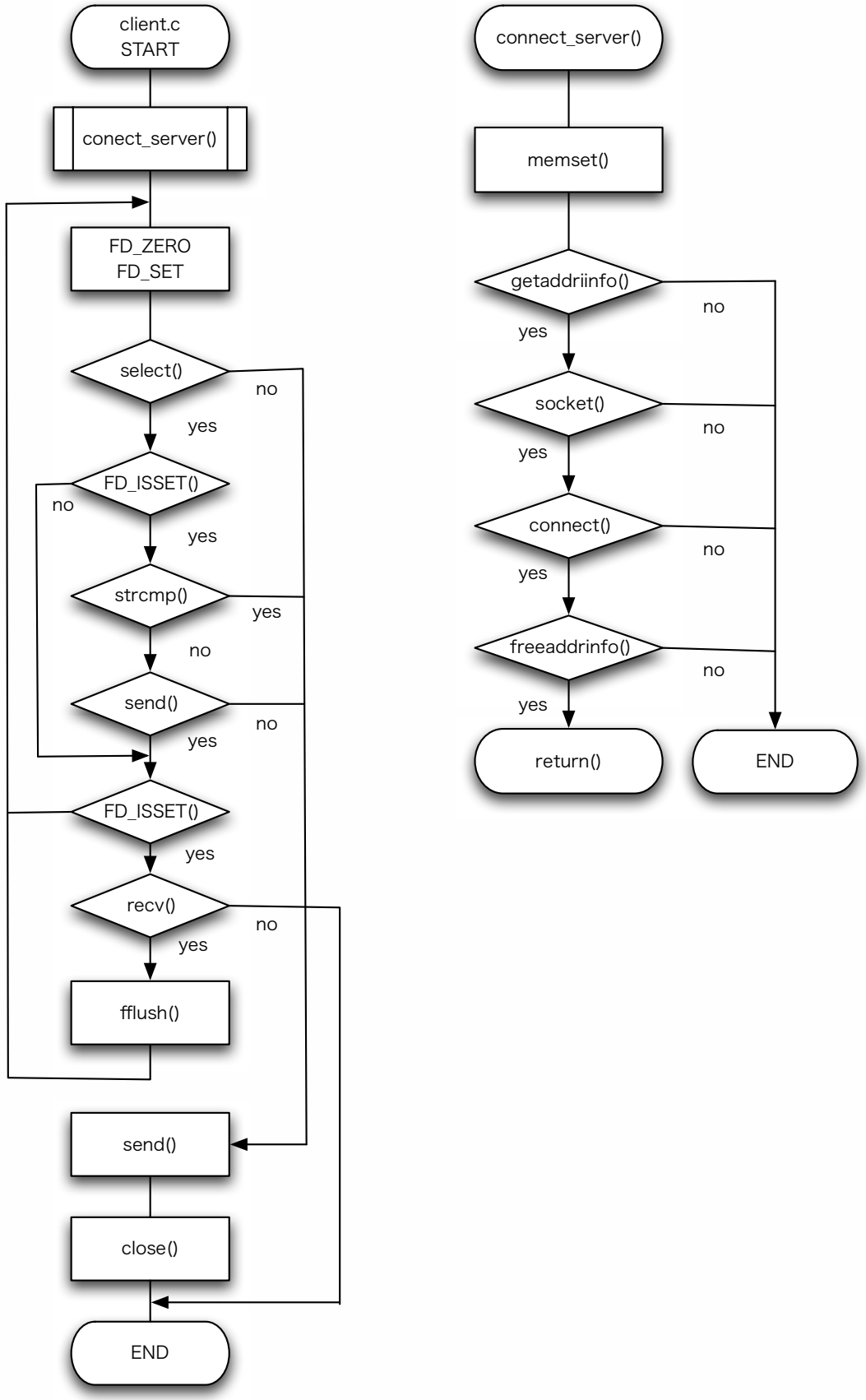

図 2: フローチャート (client.c)

2　課題2：HTTP クライアントの作成

ソケットおよび HTTP プロトコルを使って任意の WWW サーバーから任意の URL のページを取得し、標準出力に表示するコマンドプログラムを作成せよ。報告書には実行結果を示すとともに、作成したプログラムを解説せよ。

2.1　ソース (httpget.c)

ソース（httpget.c）

```
        （変更したところだけ書き出した）

15   /* check of argument */
16
17   if (argc != 3) {
18     fprintf(stderr, "Usage: %s (hostname) (/filename) \n", argv[0]);
19     exit(EXIT_FAILURE);
20   }
21   /* create socket and connect to server */
22
23   //ポート番号を８０と設定する
24   const char *port="80";
25
26   s = connect_server(argv[1], port);
27
28   /* check of input from standard input or server */
29
30   //監視対象のディスクリプタ一覧をゼロクリア
31   FD_ZERO(&readfd);
32
33   //リスニングソケットを監視対象に追加
34   FD_SET(0, &readfd);
35   FD_SET(s, &readfd);
36
37   /* input from standard input */
38
39   //"GET ./index HTTP/1.0"を送信するために、recv_buf に格納する
40   n = sprintf(recv_buf,"GET %s HTTP/1.0\n\n",argv[2]);
41
42   //サーバー側に送信を行う
43   if (send(s, recv_buf, n, 0) <= 0)
44     exit(EXIT_SUCCESS);
45
```

```
46   /* receive from server */
47   while(1){
48   //受信なので client.c と変わらない
49     if (FD_ISSET(s, &readfd)) {
50       if ((n = recv(s, recv_buf, BUFFER - 1, 0)) <= 0) {
51  fprintf(stderr, "connection closed.\n");
52  exit(EXIT_FAILURE);
53       }
54       recv_buf[n]='\0';
55       printf("%s ", recv_buf);
56       //バッファに格納されているデータを吐き出します
57       fflush(stdout);
58     }
59   }
60
61   close(s);
62
63   return(EXIT_SUCCESS);
64
65 }

        （省略）
```

2.2 実行結果 (httpget.c)

実行結果（httpget.c）

```
yoshiaki-oshiro-no-macbook-2:source yoshiaki$ ./httpget www.ie.u-ryukyu.ac.jp /index.html
connected to 'www.ie.u-ryukyu.ac.jp'
HTTP/1.1 200 OK
Date: Thu, 28 Oct 2010 18:33:36 GMT
Server: Apache/2.2.3 (CentOS)

（省略）

  </TBODY>
</TABLE>
</BODY>
</HTML>
 connection closed.
```

2.3 解説 (httpget.c)

1. ソースの説明
   (a) 17〜20 より引数が３つ (./httpget を含めて) かどうかの判断をしている
   (b) httpget.c では hostname と取得したい先の場所、ファイル名の２つの引数が必要である
   (c) 24 では始めにポート番号を設定している。
   (d) 26 では hotname とポート番号 (80) を connect_server に引数として送ってる
   (e) 40 では文字列を recv_buf に格納し、その文字数を n に入れている
   (f) 47 では while 文を使用している
   (g) 一度通るだけでは１行しか文字列が取得出来ないので、while 文を使用している
   (h) client.c の入力に関する関数はすべて削除してある
2. 実行結果の説明
   (a) www.ie.u-ryukyu.ac.jp の index.html を取得した
   (b) return キーは押さずに実行結果が表示された
   (c) 「HTTP/1.1 200 OK 」と表示されているので、エラーがなかったことがわかる

参考文献

フローチャート（流れ図）
http://wwwpat.eng.u-toyama.ac.jp/flowchart/fc_sct.html

[C 言語]Socket 間通信 echo サーバを作る
http://blog.majide.com/2009/02/socket-programming-server/

IPv6 ソケットプログラミング
http://www.nslabs.jp/socket.rhtml

C 言語 send 関数のメモと「ab」の動作
http://blog.livedoor.jp/campanella_77/archives/20319720.html

UNIX のソケット通信 (C 言語)
http://www.geocities.co.jp/Athlete-Samos/7760/study/unix_socket1.html