情報工学実験2

ネットワークプログラミング基礎　1組

担当教員名：長田 智和

実験日 2008/12/25
提出日 2008/01/30

075730G：澤岻千明

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説明せよ。

実行結果　それぞれのプログラムを実行すると、このような結果となった。

client

```
./client localhost 2600
connected to 'localhost'
TCP> Hello
Hello
TCP>
TCP> ie.u-ryukyu
ie.u-ryukyu
TCP>
TCP> quit

./client localhost 2600
connected to 'localhost'
TCP>
Timeout
```

server

```
./server 2600
receive 'Hello'
send Hello
receive ''
receive 'ie.u-ryukyu'
send ie.u-ryukyu
receive ''
receive 'quit'
send quit

./server 2600
receive 'quit'
send quit
while receiving data: Connection reset by peer
receive 'quit'
```

client 側で入力した文字列を server 側が受け取り、client 側へ返しているのがわかる。’quit’ と入力するか、Timeout で終了するようになっている。

server と client の動作を表した図 1 を以下にしめす。

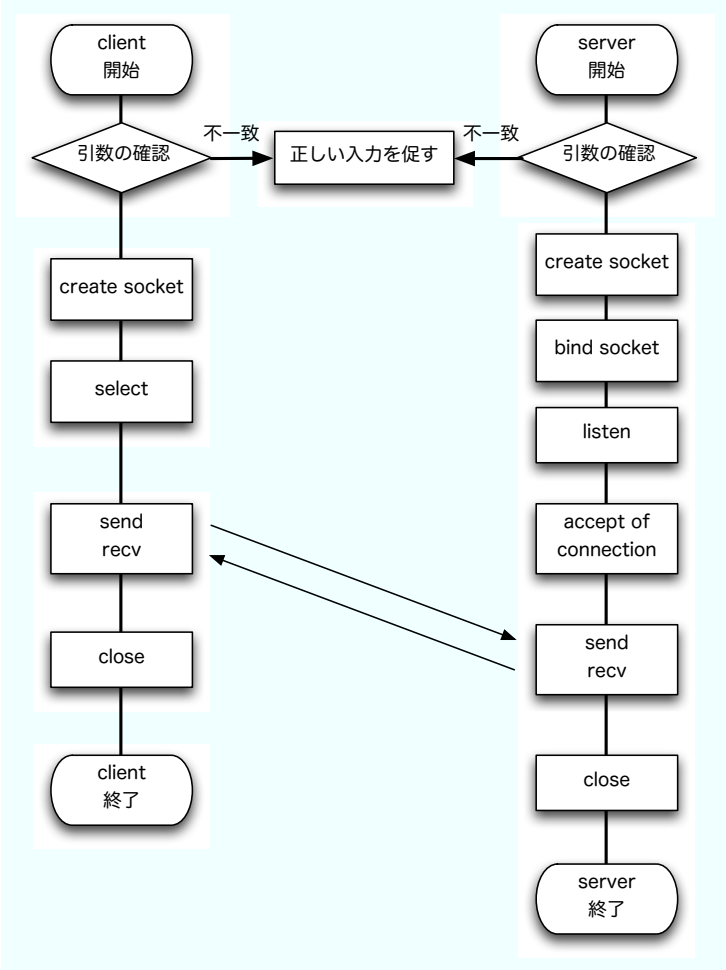

図 1: server と client のフローチャート

server.c では、まず socket 関数でソケットを生成する。ソケットとは、IP アドレスと、IP アドレスのサブアドレスであるポート番号を組み合わせたネットワークアドレスのことである。それから bind 関数でアドレス情報を結びつける。リクエストの受信待機は listen 関数で管理される。accept 関数で、接続待ちのリクエストを一つ取り出し接続を確立させて、新しいソケットを生成する。クライアントの通信はこのソケットを用いる。そして client 側からの受信、送信を行う。これらはデータを受信する recv 関数、逆に送信する send 関数によって行われる。最後は close 関数で終了する。

client.c でも、socket 関数によってソケットを作成する。その次に connect 関数で server 側へ接続する。select 関数で送受信を多重化、つまり監視し一定時間内に入力がなければ接続を終了するようにしている。server 側と同じ様にデータの送受信をし、最終的には close 関数でプログラムを終了させる。

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。

[実行例]

```
> httpget www.ie.u-ryukyu.ac.jp/index.html
```

(以下、該当ページのhtml ソースが表示される)

以下に、ソースコードを作成する上での注意事項を示す。

1. C 言語で作成すること。
2. make コマンドを使ってコンパイルできるようにMakefile を付けること。
3. 適宜コメントを付けること。

httpget.c

```
int main(int argc, char *argv[])
{

  int s;                        /* ソケットのファイルディスクリプタ */
  int n;                        /* 受信メッセージの文字数 */
  struct timeval tv;            /* タイムアウトの設定時間 */
  fd_set readfd;                /* 選択するファイルディスクリプタ */
  char recv_buf[BUFFER];        /* 受信バッファ */
  char send_buf[BUFFER];        /* 送信バッファ */
  char host_path[BUFFER];       /* ホスト名とパス */
  char host[BUFFER];            /* ホスト名 */
  char path[BUFFER];            /* パス */
  char *p;                      /* ホスト名とパスの区切り位置 */


  /* 入力確認 */
  if (argc != 2) {
    fprintf(stderr, "Usage: %s (url) \n", argv[0]);
    exit(EXIT_FAILURE);
  }

  /* ホスト名とパスの取得 */
  sscanf(argv[1], "%s", host_path);
  p = strchr(host_path, '/');
  strcpy(path, p);
  *p = '\0';
```

```
strcpy(host, host_path);

/* ソケットの生成 と サーバーへの接続 */
s = connect_server(host);

/* main loop */

while (1) {

  /* タイムアウトの時間設定 */
  tv.tv_sec  = 30;
  tv.tv_usec = 0;

  /* 標準入力、サーバーからの入力のチェック */
  FD_ZERO(&readfd);
  FD_SET(0, &readfd);

  /* タイムアウトの設定 */
  FD_SET(s, &readfd);
  if ((select(s + 1, &readfd, NULL, NULL, &tv)) <= 0) {
    fprintf(stderr, "\nTimeout\n");
    break;
  }

  /* 標準入力からの入力 */
  if (FD_ISSET(0, &readfd)) {
    char request[] = "GET / HTTP/1.0 \r\n\r\n";
    n = sizeof(request);
    strcpy(recv_buf, request);

    recv_buf[n]='\0';
    sscanf(recv_buf, "%s", send_buf);

    if (send(s, recv_buf, n, 0) <= 0)
      break;
  }

  /* サーバーからの受信 */
  if (FD_ISSET(s, &readfd)) {
    /* 受信終了の処理 */
    if ((n = recv(s, recv_buf, BUFFER - 1, 0)) <= 0) {
      fprintf(stderr, "connection closed.\n");
      exit(EXIT_FAILURE);
    }
    recv_buf[n]='\0';
```

```
      printf("%s", recv_buf);
      fflush(stdout);
    }
  strcpy(recv_buf, "quit");
  send(s, recv_buf, n, 0);
  close(s);

  return(EXIT_SUCCESS);

}

int connect_server(const char *hostname)
{
  const char *port = "80";   /* ポート番号を 80 と設定 */
  struct addrinfo hints;     /* サーバーのアドレス */
  struct addrinfo *ai;       /* アドレス情報 */
  int s;                     /* ソケットのファイルディスクリプタ  */

 -- 省略 --
```

実行結果

```
make
gcc -g   -c -o httpget.o httpget.c
gcc -g  -o httpget httpget.o

./httpget www.ie.u-ryukyu.ac.jp/index.html
connected to 'www.ie.u-ryukyu.ac.jp'

HTTP/1.1 200 OK
Date: Fri, 30 Jan 2009 09:54:41 GMT
Server: Apache/2.0.55 (Unix) mod_ssl/2.0.55 OpenSSL/0.9.7l PHP/5.1.2
Last-Modified: Thu, 29 Jan 2009 05:47:58 GMT
ETag: "1d1b0d3-7025-a16c5f80"
Accept-Ranges: bytes
Content-Length: 28709
Cache-Control: max-age=60
Expires: Fri, 30 Jan 2009 09:55:41 GMT
Connection: close
Content-Type: text/html

<HTML>
<HEAD>
```

```
<META http-equiv="Content-Type" content="text/html; charset=utf-8">
<!--<META http-equiv="Content-Style-Type" content="text/css">-->
<META http-equiv="content-style-type" content="text/css">
<LINK rel="SHORTCUT ICON" href="icon/fav-ie2.ico">

 -- 省略 --
```

引数で指定された URL のソースを出力するプログラム。HTTP で使われているポート 80 番を使う様にしている。

参考 URL

- socket-refference
  http://www.nn.osnr.jp/~nagayan/index.php?socket-refference
- Makefile の書き方 - スキルアップ輪講
  http://www.c.csce.kyushu-u.ac.jp/~seiichirou/wiki/index.php?Makefile