ネットワーク接続の残りの問題

山本昌志*

2007 年 11 月 2 日

概 要

ネットワークを接続するための，残りの問題について説明する．

1 本日の学習内容

コンピューターをネットワークに接続する場合，様々な細かい処理が必要となる．

- ホスト名や IP アドレスの取得の方法が分かる．
- `fork()` システムコールを使い，子プロセスを生成する方法が分かる．

2 もう少しましなチャットプログラム

2.1 通信を行うプログラムの例

先週のプログラムを，リスト 1 とリスト 2 のように改良した．改良の内容は，以下の通りである．

- サーバープログラムで，自分のホスト名と IP アドレスを表示できるようにした．ここでは，`gethostname()` システムコールと `gethostbyname()` 関数を使っている．
- サーバープログラムで，複数のクライアントからのメッセージを表示できるようにした．ここでは，`fork()` システムコールを使って，子プロセスを生成し，複数のクライアントに対応している．

これらの改良には，WEB ページ「UNIX ソケットプログラミング入門 (1)[1]」と「ネットワークプログラミング[2]」を参考にしている．不明な部分は，このページを見ると良いだろう．

リスト 1: テキストを送信するクライアントプログラム

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <sys/types.h>
4 #include <sys/socket.h>
5 #include <netinet/in.h>
```

*独立行政法人　秋田工業高等専門学校　電気情報工学科

[1] http://hp.vector.co.jp/authors/VA003991/kouza/senior/kouza_socket.html

[2] http://cai.int-univ.com/sugsi/Lecture/NetProg/index.html

```
6  #include <arpa/inet.h>
7
8  int main(int argc, char *argv[])
9  {
10   struct sockaddr_in server;
11   char message[80], ip_address[16];
12   int fd;
13
14   strcpy(ip_address,argv[1]);          // コマンドライン引数のIPアドレスのコピー
15   fd=socket(PF_INET, SOCK_STREAM, 0);  // ソケットの作成
16
17   memset((char *) &server, 0, sizeof(server));   // アドレス構造体の初期化
18   server.sin_family=AF_INET;
19   server.sin_port = htons(5320);
20   server.sin_addr.s_addr=inet_addr(ip_address);
21   connect(fd, (struct sockaddr *) &server, sizeof(server));
22
23
24   while(1){
25     printf("message:");
26     fgets(message, 80, stdin);
27     send(fd, message, strlen(message),0);
28     if(strncmp(message,"bye",3)==0)break;
29   }
30
31   close(fd);
32
33   return 0;
34 }
```

リスト 2: テキストを受信するサーバープログラム

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <sys/types.h>
5  #include <sys/socket.h>
6  #include <sys/socket.h>
7  #include <netinet/in.h>
8  #include <sys/wait.h>
9  #include <arpa/inet.h>
10 #include <unistd.h>
11 #include <signal.h>
12 #include <netdb.h>
13
14 #define PORT 5320
15
16 void kill_zombie(int sig);
17 void close_process(int unused);
18 char *showip(char *ip_address);
19
20 //=============================================================
21 // main 関数
22 //=============================================================
23 int main(void)
24 {
25   struct sockaddr_in client, server;
26   struct hostent *server_host;
27   int fds, fda, length;
28   pid_t pid;
29   char host_name[257];
```

```
30
31    int temp;
32
33    // ―― 自ホスト情報の取得と表示 ――――
34    bzero(host_name,257);
35    gethostname(host_name,256);                          // 自ホスト名の取得
36    server_host = gethostbyname(host_name);              // 自ホスト情報の取得
37    printf("\n――――― informations of server ――――――――\n");
38    printf("Host name:%s\n",host_name);                  // 自ホスト名の表示
39    printf("IP= %s\n",showip(server_host->h_addr));      // IPアドレスを表示
40    printf("\n\n");
41
42    fds=socket(PF_INET, SOCK_STREAM, 0);                 // ソケット作製
43
44    memset((char *) &server, 0, sizeof(server));         // ゼロで埋める
45    server.sin_family=AF_INET;
46    server.sin_addr.s_addr = inet_addr(showip(server_host->h_addr));
47    server.sin_port = htons(PORT);
48    bind(fds, (struct sockaddr *) &server, sizeof(server));  // ソケットに名前
49
50    listen(fds, 5);
51
52
53
54    length=sizeof(client);
55
56    signal(SIGINT, close_process);                       // [Ctrl]+cを押したとき
57    signal(SIGCHLD, kill_zombie);                        // 子プロセスの監視
58
59    while(1){                                            // 無限ループ
60
61      fda=accept(fds, (struct sockaddr *) &client, &length);
62
63      pid=fork();
64
65      if(pid==0){      //――――  子プロセス ――――――――――――
66        char read_str[80];                               // データ読み込み領域
67        close(fds);                                      // ソケットをクローズ
68        while(1){                                        // 無限ループ
69          memset(read_str, '\0', sizeof(read_str));      // 文字列の終わりを示す'\0'で埋める
70          recv(fda, read_str, 80, 0);                    // データの受信
71          printf("%s> ",showip((char *)&client.sin_addr));   // クライアントアドレス表示
72          printf("%s",read_str);    // 受信データーを表示
73          if(strncmp(read_str,"bye",3)==0)break;         // "bye"ならば, ループ脱出
74        }
75        close(fda);                                      // ファイルディスクリプタをクローズ
76        exit(0);                                         // 子プロセス終了
77      }else{         //――――  親プロセス ――――――――――――
78        close(fda);                                      // ファイルディスクリプタークローズ
79      }
80    }
81
82    return 0;
83  }
84
85  //=================================================================
86  // ゾンビの埋葬　以下を参考
87  // http://hp.vector.co.jp/authors/VA003991/kouza/senior/kouza_socket.html
88  //=================================================================
89  void kill_zombie(int sig){
90    while(waitpid(-1,NULL,WNOHANG)>0);
91    signal(SIGCHLD, kill_zombie);
```

```
92  }
93
94  //=============================================================
95  // [Ctrl]+cが押されたときの処理　まだ作成途中
96  //=============================================================
97  void close_process(int unused){
98    exit(0);
99  }
100
101 //=============================================================
102 // IPアドレスのポインターを返す
103 //=============================================================
104 char *showip(char *ip_address)
105 {
106   static char ip[7];                    // 静的変数．メモリーから消えない
107   char ipnum[4];
108
109   bcopy(ip_address, ipnum, 4);
110   sprintf(ip,"%u.%u.%u.%u",             // %u:符号なし10進数
111           (unsigned char)ipnum[0],
112           (unsigned char)ipnum[1],
113           (unsigned char)ipnum[2],
114           (unsigned char)ipnum[3]);
115
116   return ip;
117 }
```

2.2 自分のホスト名を取得する

自分のホスト名を取得するためには，gethostname() システムコールを使う．詳細については,「man gethostname」で調べよ．また，ホスト名からホストの情報を得るためには，gethostbyname() 関数を使う．これについても,「man gethostbyname」で調べよ．

2.3 子プロセスの生成

マルチプロセスについては，教科書 [1]pp.308–312 に詳しく書いてある．ここでは，概要を簡単に述べる．

子プロセス (実行中のプログラム) とは，親プロセスから呼び出されたプロセスである．fork() システムコールを使うと，呼び出した親プロセスと以下の部分のみが異なる子プロセスができる．

- PID(プロセス ID)．プロセスを区別する番号．
- PPID(親プロセス ID)．自分を呼び出した親プロセスのプロセス ID

fork() システムコールを実行すると，現在のプロセスのコピーが作成される．全く同じ二つのプロセスが実行されることになる．一つは親プロセス，もう一つは子プロセスである．同じプログラムではあるが，データ空間は異なる．そのため，親と子では異なった動作をさせることができる．

親と子のプロセスの区別は，fork() の戻り値で区別できる．親の戻り値は子プロセスのプロセス ID(正の整数) で，子プロセスは 0 である．fork() の戻り値が-1 の場合，子プロセスの生成の失敗を示す．

次の例が最も簡単な，子プロセスを使ったプログラムである．

```
#include <stdio.h>
#include <stdlib.h>              // exit() を使うため
#include <sys/types.h>           // fork() を使うため
#include <unistd.h>              // fork() を使うため

int main(void)
{
  int pid;
  int i=8,j=3;

  pid=fork();

  if(pid==0){     // -------- 子プロセス -------------
    printf("child process\n");
    printf("i+j=%d\n",i+j);
    exit(0);
  }else{          // -------- 親プロセス -------------
    printf("parent process\n");
    printf("i-j=%d\n",i-j);
  }

  return 0;
}
```

実行結果

```
child process
i+j=11
parent process
i-j=5
```

3 プログラム作成の練習

[**練習 1**] リスト 1 とリスト 2 を実行させて，その動作を理解せよ．

[**練習 2**] リスト 2 を参考にして，自分のホスト名と IP アドレスを表示するプログラムを作成せよ．通信の部分の記述は必要ない．

[**練習 3**] リスト 2 を参考にして，親プロセスと子プロセスで次の動作を行うプログラムを作成せよ．通信の部分の記述は必要ない．

- 親プロセスでは，6+2 を計算して，結果を表示する．
- 子プロセスでは，6-2 を計算して，結果を表示する．

[**練習 4**] `gethostbyname()` 関数を使って，`www.akita-nct.jp` のホスト情報を表示するプログラムを作成せよ．

参考文献

[1] 内田智史監修, (株) システム計画研究所編. C 言語によるプログラミング　応用編　第 2 版. (株) オーム社, 2006.