

情報工学実験 C (ソフトウェア) 報告書

## (ネットワークプログラミング)

学生番号: 09429533

提出者: 高島和嗣

提出日: 2019 年 月日 ( )

締切日: 2019 年 月日 ( )

### 1 クライアント・サーバモデルの通信の仕組み

クライアントは、サーバに要求メッセージを送信し、サーバで処理された結果を受け取り処理を行う。以下にクライアントの処理の流れを記載する。

1. 通信相手の IP アドレスの取得
2. ソケットの形成
3. 接続の確立
4. 要求メッセージを送信
5. 応答メッセージを受信
6. 応答メッセージを処理
7. ソケットの削除

サーバは、クライアントからの要求メッセージを常に待ち、要求メッセージが到着したら、処理を行い、結果を送信する。以下にサーバの処理の流れを記載する。

1. ソケットの形成
2. ソケットの名付け
3. 接続要求の待機
4. 接続要求の受理
5. 要求メッセージを受信
6. 応答メッセージを送信

## 2 名簿管理プログラムのクライアント・サーバプログラムの作成方針

今回の演習では、外部から入力されたデータを計算機で扱える内部形式に変換して格納し、それらの操作を行う名簿管理プログラムをクライアント・サーバ方式で作成した。名簿管理プログラムは、標準入力から「ID、氏名、生年月日、住所、備考」からなる CSV 形式の名簿データを受け、それらをメモリに登録する機能と標準入力から%で始まるコマンドを受け、登録してあるデータの件数を表示したり、データ自体を表示したり、整列したりする機能を持つ。名簿管理プログラムで処理を行うコマンド内容を以下に記載する。

クライアントで入力された内容をサーバに送信し、サーバ側で内容ごとに異なる処理を行うように、場

コマンド	解説	パラメータ範囲
%Q	プログラムを終了	
%C	CSV データの登録件数などを表示	
%P n	CSV データの先頭から n 件表示	n: 0 ~ 99 ( $n = 0, n > 100$ :全件表示, $n < 0$ :後ろから $-n$ 件表示)
%R file	file から読み込み	
%W file	file へ書き出し	
%F word	word を含むデータを検索	
%S n	データを n 番目の項目で整列	
%D n	n 番目のデータを削除	n: 0 ~ 99 ( $n = 0$ :全て削除, $n < 0, n > 100$ :警告文の表示)
%V word	word を含むデータを削除	
%H	コマンド一覧	
%B	データ群を 1 つ前の状態に戻す	

合分けした。クライアント側での処理も必要であると思ったコマンドはクライアント側でも場合分けした。具体的には、クライアント側で%Q、%P、%R、%W、%Fとそれ以外の場合とで分けた。

## 3 プログラムの説明

### 3.1 サーバー・クライアントプログラムの処理の流れ

- クライアント

1. gethostbyname 関数を用いて通信先の IP アドレスを獲得する。
2. socket 関数でソケットを作成し、connect 関数を用いるために sockaddr\_in 構造体を sockaddr 構造体にキャストし、connect 関数でソケットを接続する。
3. read 関数を用いて標準入力を行い、関数 parse\_line で入力内容がコマンドか、データ入力かで場合分けする。
4. コマンドなら、さらに%Q、%P、%R、%W、%Fとそれ以外の場合で処理を分ける。

- サーバ

1. socket 関数でソケットを作成し、bind 関数でソケットに名前を付け、listen 関数でクライアント側からの接続要求を待つ。

2. `accept` 関数でクライアントの接続要求を受け入れる。
3. `recv` 関数でクライアントからのメッセージを受信する。この時、メッセージが%Q ならば、`close` 関数でクライアントに接続されているファイルディスクリプタを閉じる。
4. 関数 `parse_line` で入力内容がコマンドか、データ入力かで場合分けし、コマンドの種類でも場合分けし、処理を行う。

クライアント側で、入力内容がコマンドの時に%の後に空白がない時は空白を入れるようにエラー文を表示した。また、%Q, %P, %R, %W, %F はそれぞれの関数で処理し、他のコマンドは関数 `cmd_default` で処理を行う。サーバ側では、各コマンドごとの処理を行う前に引数の型が適切であるかチェックし、適切でない時はエラー文をクライアントに送信する。

### 3.2 サーバ、クライアントのプロトコル

今回作成したプログラムのプロトコルは TCP/IP 方式である。IP とは、インターネット上の機器が持つ識別番号である IP アドレスに基づいて通信を行うプロトコルである。TCP とは、ポート番号という識別番号を用いて、IP アドレスの宛先のコンピュータのどのアプリケーションなのか識別できる。また、通信相手の状況を確認して接続を確立し、データの伝送が終わると切断する。相手がデータを受け取ったかを確認し、データの欠落や破損を検知した場合は再送したり、届いたデータを送信順に並べ直したりといった制御を行う。通信する宛先の IP アドレスが分かれば、その IP アドレス先にデータを送信できるが、どのアプリケーションでそれを受信するのか判断できない。そこで、TCP ヘッダにポート番号を付加することで、どのアプリケーションと通信するか識別する。ポート番号は、0 ~ 65535 の間で指定できるが 1023 番までのポートは、主要なプロトコルで用いられている。

### 3.3 サーバ、クライアントでのコマンドの処理内容

#### 3.4 クライアント

- コマンド Q  
関数 `cmd_quit` でサーバに入力内容%Q を送り、ソケットを閉じた後に、`exit` 関数で処理を終了する。
- コマンド P  
関数 `cmd_print` で、サーバに入力内容%P n を送り、応答メッセージを受け取る。この時、%P の引数 n が数字ではない時、応答メッセージの内容が miss となっているので、エラー文を表示する。引数が適切の場合、応答メッセージは引数に応じた表示するデータの数なので、その数の分だけループ処理をし、データを受信し表示する。
- コマンド R  
関数 `cmd_read` で、ファイルを読み取り形式で開き、引数のファイル名がクライアントに存在しない時にはエラー文を表示する。存在する場合、ファイルの中身を 1 行ずつ読み取り、CSV としてサーバに送信する。全行送り終えた後にファイルを閉じる。
- %W  
関数 `cmd_write` で、ファイルを書き込み形式で開く。サーバに入力内容%W file を送り、応答メッセージを受け取る。応答メッセージの内容はサーバに存在するデータの総数であるため、その数の分だけループ処理をし、受け取ったデータをファイルに書き込む。

- %F

関数 `cmd_find` で、サーバに入力内容 %F word を送り、応答メッセージを受け取る。応答メッセージの内容はサーバ内の引数として与えた word を含むデータの数であるため、0 の時はクライアントから送られてきたエラー文を表示し、1 以上の時はデータの数の分だけループ処理をし、受信したデータを表示する。

- %C, %S, %D, %V, %H, %B データ入力

関数 `cmd_default` でサーバに入力内容を送り、サーバから応答メッセージを受け取り表示する。また、\%C の時はサーバ内のデータの総数を記載したメッセージを受信し表示し、\%H の時は登録されているコマンド一覧を記載したメッセージを受信し表示し、\%S, \%D, \%V, \%B ではサーバ内のデータ群を変化させる処理なので、処理を完了したことを示すメッセージを受信し表示する。また、\%C, \%H は引数が必要ないので、引数が存在する時はエラー文を受信する。 \%D, \%B の引数が数字でないときもエラー文を受信する。 %S, %V は引数を含むデータがサーバに存在しない時に、引数を含むデータが存在しないことを示す文を受信する。

### 3.5 サーバ

- コマンド Q

受信した内容が %Q の時、main 部分でクライアントに接続されているファイルディスクリプタを閉じ、ループの外に出る。また、この時サーバのソケットは閉じず、新しくクライアントと接続すると処理を再開する。

- コマンド P

コマンド P の引数  $n$  を  $n > 0$ ,  $n = 0$ ,  $n < 0$  で場合分けを行い、 $n = 0$  の時は全件表示、 $n < 0$  の時は前から  $n$  件表示、 $n < 0$  の時は後ろから  $-n$  件表示し、引数が登録されているデータ数より大きいときは全件表示する。また、表示したいデータを 1 件ずつクライアントに送信するため、クライアントとの `send`, `recv` の数を同数にするためにデータを送信する前に引数の絶対値を送信し、その分だけループ処理を行い、クライアントにデータを送信する。

- コマンド R

クライアントからデータを 1 件ずつ送信されているので、サーバで %R の処理を行われない。

- %W

サーバに格納されている全てのデータをクライアント側に送信する。また、サーバとクライアントとの `send`, `recv` の数を同数にするために予め、データの総数をクライアントに送信しておく。

- %F

`strcmp_word` 関数で、引数の単語を含むデータが存在するか調べ、存在するデータの数をクライアントに送信する。データの数が 0 のときは、クライアントに引数を含むデータが存在しないことを示す文を送信し、1 以上のときは、引数の単語を含むデータを全てクライアントに送信する。

- %C

サーバに登録されているデータの数を記載した文をクライアントに送信する。

- %S

引数の値が 1 ~ 5 以外の時は、エラー文をクライアントに送信する。引数が 1 ~ 5 の時は、`compare_profile` 関数で入力された引数に応じて各メンバをソートする。比べる 2 つのデータが `compare_profile` 関数で設定した条件を満たすとき、`swap` 関数により順番を入れ替えるという操作を全データ分繰り返す。

- %D

引数が0以下もしくは登録されているデータ数以上の時は、エラー文をクライアントに送信する。引数が0の時は、全件削除する。引数  $n$  番目のデータを削除するために、 $n$  番目以降のデータを1つずつずらしていく処理を繰り返し、最後に登録されているデータの数を1少なくし、処理を完了したことを示すメッセージをクライアントに送信する。

- %V

`strcmp_word` 関数で、引数の単語を含むデータが存在するか調べ、存在するデータの数をクライアントに送信する。データが存在しない時はクライアントに引数を含むデータが存在しないことを示すメッセージをクライアントに送信し、データが存在する時はそのデータ以降のデータを1つずつずらしていく処理を繰り返しデータを削除し、処理を完了したことを示すメッセージをクライアントに送信する。

- %H

実行可能なコマンドを全て記載した文をクライアントに送信する。

- %B

データ群の形を1つ前の状態に戻すために、実行後にデータ群の形が変わるコマンド(%R, %S, %D, %V) のプログラムにそのコマンドの処理が起こる前に、1つ前のコマンドが処理した後のデータ群を別の構造体 `another[i]` に保存する。%B 実行時に構造体 `profile_data_store[i]` を `another[i]` に置き換える。

## 4 プログラムの使用法

以下にプログラムの実行時の様子を記載する。

```
%H
%%Q      : プログラムを終了
%%C      : CSV データの登録件数などを表示
%%P n    : CSV データの先頭から n 件表示 (n: 0 - 99 ... n = 0, n > 100:全件表示, n < 0:後
ろから -n 件表示)
%%R file : file から読み込み
%%W file : file へ書き出し
%%F word : word を含むデータを検索
%%S n    : データを n 番目の項目で整列
%%D n    : n 番目のデータを削除 (n: 0 - 99 ... n=0:全件削除, n < 0, n > 100:警告文の表示)
%%V word : word を含むデータを削除
%%B      : データ群の形を 1 つ前の状態に戻す
```

```
%R sample2.txt
%C
11 profile(s)
%P 3
ID      : 5100046
Name    : The Bridge
Birth   : 1845-11-02
Addr    : 14 Seafield Road Longman Inverness
```

Com. : SEN Unit 2.0 Open

ID : 5100127

Name : Bower Primary School

Birth : 1908-01-19

Addr : Bowermadden Bower Caithness

Com. : 01955 641225 Primary 25 2.6 Open

ID : 5100224

Name : Canisbay Primary School

Birth : 1928-07-05

Addr : Canisbay Wick

Com. : 01955 611337 Primary 56 3.5 Open

%D 2

%D was executed

%P 3

ID : 5100046

Name : The Bridge

Birth : 1845-11-02

Addr : 14 Seafield Road Longman Inverness

Com. : SEN Unit 2.0 Open

ID : 5100224

Name : Canisbay Primary School

Birth : 1928-07-05

Addr : Canisbay Wick

Com. : 01955 611337 Primary 56 3.5 Open

ID : 5100321

Name : Castletown Primary School

Birth : 1913-11-04

Addr : Castletown Thurso

Com. : 01847 821256 01847 821256 Primary 137 8.5 Open

%B

%B was executed

%P 3

ID : 5100046

Name : The Bridge

Birth : 1845-11-02

Addr : 14 Seafield Road Longman Inverness

Com. : SEN Unit 2.0 Open

ID : 5100127

Name : Bower Primary School

Birth : 1908-01-19  
Addr : Bowermadden Bower Caithness  
Com. : 01955 641225 Primary 25 2.6 Open

ID : 5100224  
Name : Canisbay Primary School  
Birth : 1928-07-05  
Addr : Canisbay Wick  
Com. : 01955 611337 Primary 56 3.5 Open

%F The Bridge  
ID : 5100046  
Name : The Bridge  
Birth : 1845-11-02  
Addr : 14 Seafield Road Longman Inverness  
Com. : SEN Unit 2.0 Open

%V The Bridge  
%V was executed  
%C  
10 profile(s)  
%S 2  
%S was executed  
%P 3

ID : 5100127  
Name : Bower Primary School  
Birth : 1908-01-19  
Addr : Bowermadden Bower Caithness  
Com. : 01955 641225 Primary 25 2.6 Open

ID : 5100224  
Name : Canisbay Primary School  
Birth : 1928-07-05  
Addr : Canisbay Wick  
Com. : 01955 611337 Primary 56 3.5 Open

ID : 5100321  
Name : Castletown Primary School  
Birth : 1913-11-04  
Addr : Castletown Thurso  
Com. : 01847 821256 01847 821256 Primary 137 8.5 Open

%Q

## 5 プログラムの作成過程に関する考察

ファイルの読み込み及び書き込みのコマンドを実装する際、クライアント側のファイルを指定することにした。今回作成したクライアント、サーバのプログラムは同じパソコンで実装しているが、クライアント・サーバという形式ではサーバとクライアントのコンピュータは別々で動作していることが多いと思い、その場合はサーバ側に手を加えることは難しいのではいかと思ったため、クライアント側でファイル操作することにした。また、今回実装したプログラムはファイルの文を1行ずつサーバに送信するというものであるが、ファイルごとにサーバにアップロードするという形で実装できたら、サーバ側でファイルを開くこともでき、別のクライアントからアップロードされたファイルをダウンロードすることもできると思った。

## 6 作成したプログラム

- サーバ

```
1 #include<stdio.h>
2 #include<string.h>
3 #include <stdlib.h>
4
5 #include<fcntl.h>
6 #include<netdb.h>
7 #include <netinet/in.h>
8 #include<sys/stat.h>
9 #include<sys/types.h>
10 #include<sys/socket.h>
11 #include<arpa/inet.h>
12 #include<errno.h>
13
14 #define MAX_LINE_LEN 1024
15 #define MAX_STR_LEN 69
16 #define MAX_PROFILES 10000
17
18 void parse_line(char *,int);
19
20 struct date {
21     int y;
22     int m;
23     int d;
24 };
25
26 struct profile {
27     int      id;
28     char      name[MAX_STR_LEN+1];
29     struct date birthday;
30     char      home[MAX_STR_LEN+1];
31     char      *comment;
```



```

32 };
33
34 struct profile profile_data_store[MAX_PROFILES];
35 int profile_data_nitems=0;
36 int number;
37 FILE *fp;
38 struct profile another[MAX_PROFILES];
39
40 ///%文字を別の文字に置換
41 int subst(char *str, char c1, char c2)
42 {
43     int n = 0;
44
45     while (*str) {
46         if (*str == c1) {
47             *str = c2;
48             n++;
49         }
50         str++;
51     }
52     return n;
53 }
54
55 ///%文を分割
56 int split(char *str, char *ret[], char sep, int max)
57 {
58     int n = 0;
59     ret[n++] = str;
60     while (*str && n < max) {
61         if (*str == sep){
62             *str = '\0';
63             if(*(str+1) != sep){
64                 ret[n++] = str + 1;
65             }
66         }
67         str++;
68     }
69     return n;
70 }
71
72 struct date *new_date(struct date *d, char *str)
73 {
74     char *ptr[3];
75
76     if (split(str, ptr, '-', 3) != 3){
77         return NULL;

```

```

78     } else{
79         d->y = atoi(ptr[0]);
80         d->m = atoi(ptr[1]);
81         d->d = atoi(ptr[2]);
82         return d;
83     }
84 }
85
86 struct profile *new_profile(struct profile *p, char *csv,int fd)
87 {
88     char *qtr[5];
89     if (split(csv, qtr, ',', 5) != 5) {
90         return NULL;
91     }
92     p->id = atoi(qtr[0]);
93
94     strncpy(p->name, qtr[1], MAX_STR_LEN);
95     p->name[MAX_STR_LEN] = '\0';
96
97     if (new_date(&p->birthday, qtr[2]) == NULL) {
98         return NULL;
99     }
100    strncpy(p->home, qtr[3], MAX_STR_LEN);
101    p->home[MAX_STR_LEN] = '\0';
102
103    p->comment = (char *)malloc(sizeof(char) * (strlen(qtr[4])+1));
104    strcpy(p->comment, qtr[4]);
105    return p;
106 }
107
108 ///%%コマンド処理
109 ///コマンド C
110 void cmd_check(fd)
111 {
112     char buf[MAX_LINE_LEN];
113     sprintf(buf,"%d profile(s)\n",profile_data_nitems);
114     send(fd,buf,1024,0);
115 }
116
117
118 char *date_string(char buf[],struct date *date)
119 {
120     sprintf(buf,"%04d-%02d-%02d",date->y,date->m,date->d);
121     return buf;
122 }
123

```

```

124
125 void print_profile(struct profile *p,int fd)
126 {
127     char date[20];
128     char print_buf[1024];
129     int i;
130     sprintf(print_buf,"ID      : %d\n""Name  : %s\n""Birth : %s\n""Addr  : %s\n""Com.  : %s\n",
131             //for(i=0; print_buf[i]!='\0'; i++);
132             //printf("%s",print_buf);
133             send(fd,print_buf,1024,0);
134 }
135
136 ////コマンドP
137 void cmd_print(int nitems, int fd)
138 {
139     int i, start = 0,end = profile_data_nitems;
140     char num[1024];
141     char str[1024];
142
143     if (nitems > 0) {
144         if(nitems < profile_data_nitems) end = nitems;
145         sprintf(num,"%d",nitems);
146         send(fd,num,1024,0);
147     }
148     else if (nitems < 0) {
149         if(end + nitems > start) start = end + nitems ;
150         sprintf(num,"%d",-nitems);
151         send(fd,num,1024,0);
152     } else {
153         sprintf(num,"%d",profile_data_nitems);    //全件表示
154         send(fd,num,1024,0);
155     }
156
157     for (i = start; i < end; i++) {
158         recv(fd,str,1024,0);
159         print_profile(&profile_data_store[i],fd);
160     }
161 }
162
163 /*
164 //コマンドR
165 void cmd_read(char *filename,int fd)
166 {
167     int l;
168     char line[MAX_LINE_LEN + 1];
169

```

```

170  if((fp = fopen(filename,"r")) == NULL){
171      fprintf(stderr, "%R: file open error %s.\n", filename);
172      return;
173  }
174
175  /////%D用/////
176      number=profile_data_nitems;
177      for(l=0; l<=number-1; l++){
178          another[l] = profile_data_store[l];
179      }
180      while(get_line(fp,line)){
181          parse_line(line,fd);
182      }
183      fclose(fp);
184  }
185  */
186
187  ////コマンド C
188  void fprint_profile_csv(int fd,struct profile *p)
189  {
190      char buf[1024];
191      char date[20];
192      sprintf(buf,"%d,%s,%s,%s,%s",p->id,p->name,date_string(date,&p->birthday),p->home,p->cor
193      send(fd,buf,1024,0);
194  }
195
196  ////コマンド W
197  void cmd_write(char *filename,int fd)
198  {
199      int i,l;
200      char num[10],str[10];
201      if((fp = fopen(filename,"w")) == NULL){
202          fprintf(stderr,"file error");
203      }
204      sprintf(num,"%d",profile_data_nitems);
205      send(fd,num,10,0);
206
207      number=profile_data_nitems;
208      for(i=0; i<=number-1; i++){
209          another[i] = profile_data_store[i];
210      }
211
212      for (i = 0; i < profile_data_nitems; i++) {
213          recv(fd,str,10,0);
214          fprint_profile_csv(fd,&profile_data_store[i]);
215      }

```

```

216  fclose(fp);
217 }
218
219
220 int strcmp_word(struct profile *p,char *word)
221 {
222     char id[20];
223     char date[20];
224     sprintf(id,"%d",p->id);
225     if(strcmp(id, word) == 0||strcmp(p->name, word) == 0||strcmp(date_string(date, &p->birth), word) == 0){
226         return 0;
227     }
228 }
229
230 ///コマンドF
231 void cmd_find(char *word,int fd)
232 {
233     int i,times=0;
234     struct profile *p,*q;
235     char num[10],buf[1024];
236     char str[10];
237
238     memset(buf,0,MAX_LINE_LEN);
239
240     for (i=0; i < profile_data_nitems; i++) {
241         p=&profile_data_store[i];
242         if(strcmp_word(p,word)==0){
243             times++;
244         }
245     }
246     sprintf(num,"%d",times);
247     send(fd,num,10,0);
248     if(times>0){
249         for (i=0; i < profile_data_nitems; i++) {
250             q=&profile_data_store[i];
251             if(strcmp_word(q,word)==0){
252                 recv(fd,str,10,0);
253                 print_profile(q,fd);
254             }
255         }
256     }else{
257         //times==0 : %F の引数を含むデータがないとき
258         recv(fd,str,10,0);
259         sprintf(buf,"no data with that phrase!\n");
260         send(fd,buf,1024,0);
261     }
262 }

```

```

262
263
264 void swap(struct profile *p1, struct profile *p2)
265 {
266     struct profile tmp;
267
268     tmp = *p1;
269     *p1 = *p2;
270     *p2 = tmp;
271 }
272
273 int compare_date(struct date *d1, struct date *d2)
274 {
275     if (d1->y != d2->y) return (d1->y) - (d2->y);
276     if (d1->m != d2->m) return (d1->m) - (d2->m);
277     return (d1->d) - (d2->d);
278 }
279
280 //引数でソートを場合分け
281 int compare_profile(struct profile *p1, struct profile *p2, int column)
282 {
283     switch (column) {
284         case 1:
285             return p1->id - p2->id; break;
286         case 2:
287             return strcmp(p1->name,p2->name); break;
288         case 3:
289             return compare_date(&p1->birthday,&p2->birthday); break;
290         case 4:
291             return strcmp(p1->home,p2->home); break;
292         case 5:
293             return strcmp(p1->comment,p2->comment); break;
294     }
295 }
296
297 ///コマンドs
298 void cmd_sort(int column,int fd)                //エラー処理不十分
299 {
300     int length =profile_data_nitems;
301     int i,j,l,s;
302     struct profile *p;
303     char buf[1024],num[10];
304     memset(buf,0,MAX_LINE_LEN);
305     s = length-1;
306
307     if(column != 1 && column != 2 && column != 3 && column != 4 && column !=5){

```

```

308     sprintf(buf,"%d is not adaptted\n",column);
309     send(fd,buf,1024,0);
310 }
311 if(column == 1 || column == 2 || column == 3 || column == 4 || column ==5){
312     number=profile_data_nitems;
313     for(l=0; l<=number-1; l++){
314         another[l] = profile_data_store[l];           //コマンド D 用
315     }
316     for(i = 0; i <= s; i++) {
317         for (j = 0; j <= s - 1; j++) {
318             p=&profile_data_store[j];
319             if (compare_profile(p, (p+1), column) > 0)
320                 swap(p, (p+1));
321         }
322     }
323     sprintf(buf,"%S was executed\n");
324     send(fd,buf,1024,0);
325 }
326 }
327
328 ////コマンド D
329 void cmd_delete(int nitems,int fd)
330 {
331     int i, l, end = profile_data_nitems-1;
332     char buf[1024];
333     memset(buf,1024,0);
334     if(nitems < 0){
335         sprintf(buf,"%d is smaller than 0\n",nitems);
336         send(fd,buf,1024,0);
337     }
338     else if(nitems > end+1){
339         sprintf(buf,"%d is bigger than data number\n",nitems);
340         send(fd,buf,1024,0);
341     }
342     else{
343         number=profile_data_nitems;
344         for(l=0; l<=number-1; l++){
345             another[l] = profile_data_store[l];           //コマンド D 用
346         }
347         if(nitems > 0 && nitems < end+1){
348             for(i=nitems-1;i<end;i++){
349                 profile_data_store[i]=profile_data_store[i+1];
350             }
351             profile_data_nitems--;
352         }
353         else if(nitems == end+1){

```

```

354         profile_data_nitems--;
355     }
356     else if(nitems == 0){           //ALL DELETE
357         profile_data_nitems=0;
358     }
359     sprintf(buf,"%D was executed\n");
360     send(fd,buf,1024,0);
361 }
362 }
363
364 ////コマンド V
365 void cmd_vanish(char *word,int fd)
366 {
367     int i,k,l,times=0;
368     char date[20];
369     char id[20];
370     char buf[1024];
371     memset(buf,1024,0);
372     struct profile *p;
373     for (i=0; i < profile_data_nitems; i++) {
374         p=&profile_data_store[i];
375         if(strcmp_word(p,word)==0){
376             times++;
377         }
378     }
379     if(times>0){
380         number=profile_data_nitems;
381         for(l=0; l<=number-1; l++){
382             another[l] = profile_data_store[l];           //コマンドD用
383         }
384         for (i=0; i < profile_data_nitems; i++) {
385             p=&profile_data_store[i];
386             if(strcmp_word(p,word)==0){
387                 for(k=i;k<profile_data_nitems-1;k++){
388                     profile_data_store[k]=profile_data_store[k+1];
389                 }
390                 profile_data_nitems--;
391             }
392         }
393         sprintf(buf,"%V was executed\n");
394         send(fd,buf,1024,0);
395     } else {
396         sprintf(buf,"no data with that phrase!\n");
397         send(fd,buf,1024,0);
398     }

```



```

399 }
400
401
402 void cmd_help(int fd)
403 {
404     char help_buf[1024]={"\0"};
405     int i;
406     char *str1 = "%%Q      : プログラムを終了\n";
407     char *str2 = "%%C      : CSV データの登録件数などを表示 \n";
408     char *str3 = "%%P n    : CSVデータの先頭からn件表示 (n: 0 - 99 ... n = 0,
n > 100:全件表示, n < 0:後ろから-n件表示)\n";
409     char *str4 = "%%R file : file から読み込み \n";
410     char *str5 = "%%W file : file へ書き出し \n";
411     char *str6 = "%%F word : word を含むデータを検索\n";
412     char *str7 = "%%S n    : データを n 番目の項目で整列\n";
413     char *str8 = "%%D n    : n 番目のデータを削除 (n: 0 - 99 ... n=0:全件
削除, n < 0, n > 100:警告文の表示)\n";
414     char *str9 = "%%V word : word を含むデータを削除\n";
415     char *str10="%%B      : データ群の形を 1 つ前の状態に戻す\n";
416     sprintf(help_buf,"%s%s%s%s%s%s%s%s\n",str1,str2,str3,str4,str5,str6,str7,str8,str9);
417
418     for(i=0; help_buf[i]!='\0'; i++);
419     send(fd,help_buf,i,0);
420 }
421
422
423 void cmd_back(int fd){
424     char buf[1024];
425     int i,s=profile_data_nitems;
426     profile_data_nitems=number;
427     for(i=0; i<=profile_data_nitems-1; i++){
428         profile_data_store[i] = another[i];
429     }
430     sprintf(buf,"%B was executed\n");
431     send(fd,buf,1024,0);
432 }
433
434
435 int check1(char *param){
436 if((*param>='a'&& *param<='z') || (*param>='A' && *param<='Z') || (*param>='0' && *param<='9')){
437     return 1;
438 }
439 }
440
441 int check2(char *param){
442 if(*param>='0' && *param<='9') {

```

```

443     return 1;
444 }
445 }
446
447 int check3(char *param){
448     int l,i,j=0;
449     for(l=0;param[l]!='\0';l++);
450
451     for(i=0;i<l;i++){
452         if((param[i]>='a'&& param[i]<='z') || (param[i]>='A' && param[i]<='Z')){
453             j++;
454         }
455     }
456     return j;
457 }
458
459 /*
460 int check3(char *param){
461     if((*param>='a'&& *param<='z') || (*param>='A' && *param<='Z')) {
462         return 1;
463     }
464 }
465 */
466 void exec_command(char cmd, char *param, int fd)
467 {
468     char buf[1024];
469     memset(buf,0,MAX_LINE_LEN);
470     switch (cmd){
471     case 'C':
472         if(check1(param)==1) {
473             sprintf(buf,"Don't write word\n");
474             send(fd,buf,1024,0);
475             break;
476         }else{
477             cmd_check(fd);
478             break;
479         }
480     case 'P':
481         if(check3(param)!=0) {
482             sprintf(buf,"miss");
483             send(fd,buf,1024,0);
484             break;
485         }else{
486             cmd_print(atoi(param),fd);
487             break;
488         }

```

```

489     case 'W': cmd_write(param,fd);          break;
490     case 'F': cmd_find(param,fd);          break;
491     case 'S':
492         if(check3(param)!=0) {
493             sprintf(buf,"Please write number\n");
494             send(fd,buf,1024,0);
495             break;
496         }else{
497             cmd_sort(atoi(param),fd); break;
498         }
499     case 'D':
500         if(check3(param)!=0) {
501             sprintf(buf,"Please write number\n");
502             send(fd,buf,1024,0);
503             break;
504         }else{
505             cmd_delete(atoi(param),fd);
506             break;
507         }
508     case 'V': cmd_vanish(param,fd);          break;
509
510     case 'H':
511         if(check1(param)==1) {
512             sprintf(buf,"Don't write word\n");
513             send(fd,buf,1024,0);
514             break;
515         }else{
516             cmd_help(fd);
517             break;
518         }
519     case 'B':
520         if(check1(param)==1) {
521             sprintf(buf,"Don't write word\n");
522             send(fd,buf,1024,0);
523             break;
524         }else{
525             cmd_back(fd);
526             break;
527         }
528     default:
529         sprintf(buf,"command %c is ignored.\n",cmd);
530         send(fd,buf,1024,0);
531         break;
532 }
533 }
534

```

```

535 ///%%parse_line
536 void parse_line(char *line,int fd)
537 {
538     int x=0;
539     char buf[3];
540     memset(buf,0,3);
541     if(line[0] == '%'){
542         exec_command(line[1],&line[3],fd);
543     }
544     else{
545 /*
546         if(new_profile(&profile_data_store[profile_data_nitems], line,fd)==NULL){
547             send(fd,"error",10,0);
548         } else{
549 */
550             new_profile(&profile_data_store[profile_data_nitems], line,fd);
551             send(fd,"ok",3,0);
552             profile_data_nitems++;
553             //}
554         }
555     }
556
557 int main (){
558     struct sockaddr_in read_sa;
559     struct sockaddr_in write_sa;
560     struct hostent *host;
561     int sockfd,new_sockfd;
562     int buf_len,write_len,i;
563     char recv_buf[1024],send_buf[1024];
564     int x=1;
565     memset(recv_buf,0,MAX_LINE_LEN);
566     memset(send_buf,0,MAX_LINE_LEN);
567
568 /*make socket*/
569     if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0){
570         fprintf(stderr,"socket error\n");
571         return 1;
572     }
573
574 /*socket setting*/
575     memset((char*)&read_sa,0,sizeof(read_sa));
576     read_sa.sin_family      = AF_INET; // host address type
577     read_sa.sin_port        = htons(3001); // port number
578     read_sa.sin_addr.s_addr = htonl(INADDR_ANY);
579
580     if (bind(sockfd, (struct sockaddr *)&read_sa, sizeof(read_sa)) < 0){

```

```

581         fprintf(stderr,"bind error\n");
582         return 1;
583     }
584
585     if(listen(sockfd,5) < 0 ){
586         fprintf(stderr,"listen error\n");
587         close(sockfd);
588         return 1;
589     }
590
591     while(1){
592         if((new_sockfd = accept(sockfd,(struct sockaddr*)&write_sa,&write_len))== -1){
593             fprintf(stderr,"accept error\n");
594             return 1;
595         }
596         while(1){
597             recv(new_sockfd,recv_buf,1024,0);
598
599             //コマンド Q の処理
600             if(recv_buf[0]=='%' && recv_buf[1]=='Q'){
601                 sprintf(send_buf,"exit!");
602                 send(new_sockfd,send_buf,1024,0);
603                 close(new_sockfd);
604                 break;
605             }
606
607             parse_line(recv_buf,new_sockfd);
608         }
609     }
610 }

```

- クライアント

```

1  #include<stdio.h>
2  #include<string.h>
3  #include <stdlib.h>
4
5  #include<netdb.h>
6  #include<sys/types.h> /* 「注意」 参照 */
7  #include<sys/socket.h>
8  #include<arpa/inet.h>
9  #include <fcntl.h>
10 #include<unistd.h>
11
12 #define MAX_LINE_LEN 1024
13 FILE *fp;

```

```

14 struct hostent *hp;
15 struct sockaddr_in sa;
16
17 int sockfd;
18 void parse_line(char *);
19
20 int check1(char *param){
21     if((*param>='a'&& *param<='z') || (*param>='A' && *param<='Z') ||
22         (*param>='0' && *param<='9')) {
23         return 1;
24     }
25 }
26
27 int subst(char *str, char c1, char c2)
28 {
29     int n = 0;
30
31     while (*str) {
32         if (*str == c1) {
33             *str = c2;
34             n++;
35         }
36         str++;
37     }
38     return n;
39 }
40
41 int split(char *str, char *ret[], char sep, int max)
42 {
43     int n = 0;
44     ret[n++] = str;
45     while (*str && n < max) {
46         if (*str == sep){
47             *str = '\0';
48             if(*(str+1) != sep){
49                 ret[n++] = str + 1;
50             }
51         }
52         str++;
53     }
54     return n;
55 }
56
57 int get_line(FILE *fp, char *line)
58 {
59     if (fgets(line, MAX_LINE_LEN + 1, fp) == NULL){

```

```

60     return 0;
61 } else{
62     subst(line, '\n', '\0');
63     return 1;
64 }
65 }
66
67 //コマンド Q
68 void cmd_quit(char *line)
69 {
70     char buf[1024];
71     send(sockfd,line,MAX_LINE_LEN,0);
72     recv(sockfd,buf,MAX_LINE_LEN,0);
73     close(sockfd);
74     exit(0);
75 }
76
77 //コマンド P
78 void cmd_print(char *line){
79     int i,nitems;
80     char num[1024],buf[MAX_LINE_LEN];
81     memset(buf,0,MAX_LINE_LEN);
82     memset(num,0,1024);
83
84     send(sockfd,line,MAX_LINE_LEN,0);
85     recv(sockfd,num,1024,0);
86
87     if(strcmp(num,"miss")==0){
88         printf("Please write number\n");
89     }else{
90         nitems=atoi(num);
91
92         for(i=0;i<nitems;i++){
93             send(sockfd,"do",10,0);
94             recv(sockfd,buf,MAX_LINE_LEN,0);
95             printf("%s",buf);
96         }
97     }
98 }
99
100 //コマンド R
101 void cmd_read(char *filename){
102     char read_buf[MAX_LINE_LEN],recv_buf[MAX_LINE_LEN];
103     int i,buf_len,x=1;
104
105     if((fp = fopen(filename,"r")) == NULL) {

```

```

106         fprintf(stderr, "%sR: file open error %s.\n", filename);
107         return;
108     }
109
110     memset(read_buf,0,MAX_LINE_LEN);
111     while(fgets(read_buf,MAX_LINE_LEN,fp) != NULL){
112         //printf("%s",read_buf); //確認用
113         send(sockfd,read_buf,MAX_LINE_LEN,0);
114         recv(sockfd,recv_buf,MAX_LINE_LEN,0);
115         memset(read_buf,0,MAX_LINE_LEN);
116     }
117
118     fclose(fp);
119 }
120
121 //コマンド W
122 void cmd_write(char *line,char *filename){
123     int i,nitems;
124     char num[10],buf[1024];
125
126     if((fp = fopen(filename,"w")) == NULL){
127         fprintf(stderr,"error");
128         return ;
129     }
130
131     memset(buf,0,MAX_LINE_LEN);
132     memset(num,0,10);
133
134     send(sockfd,line,MAX_LINE_LEN,0);
135     recv(sockfd,num,10,0);
136     nitems=atoi(num);
137
138     for (i = 0; i < nitems; i++) {
139         send(sockfd,"do",10,0);
140         recv(sockfd,buf,MAX_LINE_LEN,0);
141         fprintf(fp,"%s",buf);
142     }
143     fclose(fp);
144 }
145
146 //コマンド F
147 void cmd_find(char *line,char *param){
148     char recv_buf[MAX_LINE_LEN],num[10];
149     int i,k,times;
150     memset(recv_buf,0,MAX_LINE_LEN);
151

```



```

152     for(i=0; line[i] != '\0'; i++);
153     send(sockfd,line,i,0);
154     recv(sockfd,num,10,0);
155     times=atoi(num);
156     if(times==0) {
157         send(sockfd,"miss",10,0);
158         recv(sockfd,recv_buf,1024,0);
159         printf("%s",recv_buf);
160     }
161     for(i=0;i<times;i++){
162         send(sockfd,"do",10,0);
163         recv(sockfd,recv_buf,1024,0);
164         printf("%s",recv_buf);
165     }
166 }
167
168 //コマンド デフォルト (%S,%D,%V,%H)
169 void cmd_default(char *line){
170     char recv_buf[MAX_LINE_LEN];
171     int buf_len,i;
172     memset(recv_buf,0,MAX_LINE_LEN);
173     //for(i=0; line[i] != '\0'; i++);
174     send(sockfd,line,1024,0);
175     recv(sockfd,recv_buf,1024,0);
176     if(line[0]=='%'){
177         printf("%s",recv_buf);
178     }
179 }
180
181 void exec_command(char cmd, char *param,char *line){
182     int i,buf_len;
183     char buf[1024];
184
185     switch (cmd){
186     case 'Q': cmd_quit(line);                break;
187     case 'P': cmd_print(line);                break;
188     case 'R': cmd_read(param);                break;
189     case 'W': cmd_write(line,param);          break;
190     case 'F': cmd_find(line,param);           break;
191     default:
192         cmd_default(line);                    break;
193     }
194 }
195
196 void parse_line(char *line)
197 {

```

```

198     int i,buf_len;
199     char buf[MAX_LINE_LEN];
200     if(line[0] == '%'){
201         if(check1(&line[2])==1) {
202             printf("Please space after %c%c\n",line[0],line[1]);
203         }
204         else {
205             exec_command(line[1],&line[3],line);
206         }
207     }
208     else{
209         cmd_default(line);
210     }
211 }
212
213 void send_input(){
214     char buf[MAX_LINE_LEN];
215
216     int i;
217     memset(buf,0,MAX_LINE_LEN);
218     read(0,buf,MAX_LINE_LEN);
219     //printf("%s",buf);        //入力確認
220     subst(buf, '\n', '\0');
221     parse_line(buf);
222 }
223
224 int make_sockfd(){
225     int fd;
226     if ((fd = socket(AF_INET, SOCK_STREAM, 0)) < 0){
227         fprintf(stderr,"socket error\n");
228         return -1;
229     }
230
231     //memset(&sa,0,sizeof(sa));
232     sa.sin_family      = AF_INET; // host address type
233     sa.sin_port        = htons(3001); // port number
234     bzero((char*)&sa.sin_addr,sizeof(sa.sin_addr));
235     memcpy((char*)&sa.sin_addr,(char*)hp->h_addr,hp->h_length);
236
237     if(connect(fd, (struct sockaddr*)&sa, sizeof(sa)) < 0){
238         fprintf(stderr,"connect error\n");
239         return -1;
240     }
241     return fd;
242 }
243

```

```
244 int main (){
245     hp = gethostbyname("localhost");
246
247     if (hp==NULL){
248         fprintf(stderr,"ホスト取得失敗\n");
249         return 1;
250     }
251
252     if((sockfd=make_sockfd()) < 0) return;
253
254     while(1){
255         send_input();
256     }
257 }
258
```