情報工学第三 期末レポート

氏名: 木下直樹
学籍番号: 09425521

提出日: 2015 月 11 月 26 日
締切日: 2015 年 11 月 26 日

1　クライアント , サーバモデルの通信の仕組みについて

クライアントサーバモデルとは, サービスの役割をクライアントとサーバに分離して運用することによってコンピュータの処理を分散する仕組みである. サーバはデータベースやサービスを提供するための処理を集中管理し , クライアントは極力データの送受信に専念することで, クライアント側のコンピュータの能力では処理できない, もしくは処理が遅い等の問題が消化される.

今回の実験では, 通信プロトコルとして TCP/IP を利用したプログラムを作成した. 以下に TCP/IP でのネットワークの接続やメッセージのやりとりの簡単な流れを使用するシステムコールを示す.

1.1　クライアントプログラム

1. socket() により, サーバと接続するソケットを作成する.
2. ポート番号, IP アドレスを指定するための構造体を設定する.
3. connect() により, サーバとの接続の確立する.
4. send(),recv() によるデータの送受信する.
5. close() でソケットの切断する.

1.2　サーバプログラム

1. socket() により, 接続をリスニングするソケットを作成する.
2. ポート番号, IP アドレスを指定するための構造体を設定する.
3. bind() により, ソケットにポート番号, IP アドレスを設定する.
4. listen() でソケットの接続準備する.
5. accept() でソケットの接続待機する.
6. send(),recv() によるデータの送受信する.
7. close() でソケットの切断する.

2　自由課題プログラムの作成方針

今回の自由課題では英単語暗記プログラムを作成する.

具体的には, サーバから単語の日本語訳をクライアントへ送信し, 英語訳をクライアント側の端末で入力してサーバに送信する. サーバはその正誤を判断し, 結果をクライアントへ送るというものだ.

また, 一連の処理に加え, 問題ファイルを追加する機能と, アカウント機能を追加する. このアカウント機能はログインしたクライアントが間違った問題を出力するためのものである.

3　プログラムについて

3.1　プログラムの処理の流れ

3.1.1　初期状態の処理

プログラムを実行すると, 次の様にサーバからのメッセージが届く.

```
コマンドを入力してください
practice
read
new account
log in
quit
```

この状態では practice, read, new account, log in, quit の 5 つのコマンドが入力できる. それぞれの入力に対する処理をサーバが行う. また, それ以外の入力をサーバに送信した場合, サーバは上記のメッセージを送信する処理部へループし, クライアント側のプログラムは上記のメッセージを受け取る処理部へループする.

クライアントプログラムの処理の流れは以下である.

```
(ループ){
   (ループ){
      (ループ){
         コマンドの入力
         上記の 5 つのコマンドのいずれかの入力でループから離脱
       }
       read, new account, log in の処理
      コマンド入力時に practice を入力した場合ループ離脱
   }
   practice の処理
}
```

read, new account, log in のコマンド処理は一塊のループで行い, practice はそのループの外で処理をする.

サーバプログラムの場合はクライアントからのコマンドを受け取ると, 正しいコマンドである場合その処理をする関数へとび, 正しくないコマンドを受け取った場合, クライアントがループして

初期メッセージを受け取れるようにこちらもループをして再度初期メッセージを送信する. ただし quit コマンドを受け取るとクライアントへメッセージを送信することなくプロセスは終了する.
サーバプログラムの処理の流れは以下である.

```
(ループ){
   (ループ){
      コマンド入力要求
        正しいコマンドでループ離脱
   }
   各コマンドの処理関数へ入る
}
```

3.1.2 log in 後の処理

log in コマンドを実行するとログインフラグの値を変更し, ログイン名を格納する配列にその文字列を格納する. また, コマンド入力要求が以下に変わる.

コマンドを入力してください
practice
read
review
log out
quit

この状態で practice を実行すると, 間違えた問題が account ディレクトリ内にあるアカウント名のディレクトリ内の mislog.csv ファイルへ出力する (例えば, ./account/naoki/mislog.csv). log out で初期状態のコマンド要求部と同じ処理へ戻る.

3.2 プロトコルについて

サーバとクライアント間のプロトコルで決められたコマンドとそのコマンドに対するサーバの挙動について説明する.

- practice
  戻り値:[\n 練習ファイル一覧 \n(.csv ファイル名一覧)]
  送られてきた.csv ファイル名を選んで入力し, 送信する. サーバは送られてきたファイル名が正しければそのファイルを読み込み, 問の文字列をクライアントへ送信する. クライアントはそれに対する答えをサーバへ送信する. サーバは結果の正誤と次の問をクライアントへ送信する. その送受信を繰り返し. 最後の問の正誤と同時に正解の回答数の結果を送信する. また, log in の状態で practice を実行する時, ログインしたアカウントのディレクトリ内の mislog.csv ファイルを開き, 誤回答がある度にその問題と答えをファイルへ出力する.

- read
  戻り値:[追加するファイル名を入力してください]
  クライアントは practice で使用したいファイル名をサーバへ送る. クライアントプログラム

はそのファイルを読み込みサーバへ送信する．サーバは受信した文字列を read ディレクトリ内に開いたファイルへ出力する．クライアント側にないファイルの名前を送信した場合も正常にファイルが読み込めた場合もコマンド要求部へ戻る．

- new account
  戻り値:[アカウント名を入力してください]
  作成したいアカウント名を入力し, 送信する．サーバは既にそのアカウントが存在したり, そのアカウントが作成できない場合, 文字列 [このアカウントは使えません] を送信する．アカウントが作成できる場合, 文字列 [アカウントを作成しました] を送信する．このプログラムでいうアカウントの作成とは, account ディレクトリ内にそのアカウント名のディレクトリを作成することである．

- log in
  戻り値:[アカウント名を入力してください]
  クライアントが送信するアカウント名が存在するアカウントの名前でなければ, サーバは文字列 [アカウント名が間違っています] をクライアントへ送信する．アカウントが存在する場合は, 文字列 [ログイン：(アカウント名)] をクライアントへ送信する．この際, サーバはログイン状態を示すフラグを更新し, アカウント名を格納する変数にそのアカウント名を格納する．クライアントプログラムはログインできたことを確認するとサーバ同様ログイン状態を示すフラグを更新する．

- log out
  戻り値:[ログアウトしました]
  サーバとクライアントのプログラムはログイン状態を示すフラグを更新する．また, サーバプログラムはアカウント名変数を初期化する．

- review
  フラグ変数 rflag を更新し, practice へ入る．review コマンドの入力から practice へ入る場合, ファイル名の入力要求はされず, アカウントディレクトリ内の mislog.csv ファイルを読み込む．

- quit
  クライアントはこのコマンドを送信し, プログラムを終了する．サーバはこのコマンドを受信するとプログラムを終了する．

4 プログラムの使用法

- ログインした場合もしていない場合も問題演習をする場合は practice コマンドを入力し, 各問題に対する答えを入力させていく．

- 演習で間違えた問題だけを集めた演習をしたい時は, あらかじめ new account コマンドを実行してアカウントを作成しておく．

- log in コマンドを実行してログインした状態で問題演習をすると間違えた問題が保持され, review コマンドを実行することでその問題を演習することができる．

- また, このファイルのデータはプログラムを終了しても次の practice の実行 (ログイン時) まで保持される.
- log out コマンドでログアウトができる. アカウントを切り替える際は一度ログアウトを実行して別のアカウント名でログインするとよい.
- プログラムを終了する際は quit コマンドを入力する.

5 プログラムの作成過程と成果物に関する考察

5.1 作成過程に関する工夫

5.1.1 出力管理について

本プログラムはデータの送受信をするため, 自身のプログラム自体にエラーがなくコンパイルが通ってもいざ実行すると互いの send や recv に関するエラーやバグが発生する. そのため, 本プログラムでは send と recv を関数で管理し, 実行する引数によって送受信した文字列を txt ファイルやターミナルに出力できるようにした.

ターミナルへの出力に関してはデータの壊れる瞬間や想定していない送受信を感知することに非常に役立ったが, txt ファイルに出力する際は実行を中断するとファイルに正しく書き込まれないことがあったり処理の調整が上手くいかなかったりと, 今回のプログラミングではあまり役にたたなかった.

ターミナルへ出力する際は元々出力される様になっている文字列がデバック時に邪魔となったので, printf も関数や define などを使用して管理し, 出力の ON と OFF を切り替え出きるようにすればよかったと思う.

5.1.2 バージョン管理について

本プログラムを作成する際, 小さな改良を目標としたバージョンアップの積み重ねでプログラムを大きくしていくことを心がけた. プログラムを書き換えると高い確立でエラーやバグが発生する. それを改善できればよいが, エラーチェックのための出力や処理の変更により手に負えない状態に陥ることがあるため, プログラムの書き換えをする際にはまず現状のプログラムを別のディレクトリへ退避させ, いつでもその状態へ戻れるようにした.

常にバックアップをとることでプログラムの書き換えに多少の冒険ができ, 精神的負荷と時間の負荷を軽減することに役立った. しかし, このようなバックアップの管理などはバージョン管理システムなどを利用することが普通であり, それを使いこなせるようになればプログラムを書く際に有利であるため, 次にプログラムを作成する際はこれを利用するべきであると感じた.

5.2 成果物に関する工夫

本プログラムでは機能の幅を広げやすくなるように実行プログラムのあるディレクトリ内のファイルだけでなく, 別ディレクトリの中を参照したり, ディレクトリを作成したりといった動作を使用した.

例えば, new account コマンドでは, account ディレクトリ内にアカウント名のディレクトリを作成する. これにより, 他のアカウントでのログイン時や非ログイン時の操作に影響を与えないエリアを確保できる. 本プログラムでは実装していないが, 例えば以下のような拡張ができる.

- ログイン時の read で自分のディレクトリ内にファイルを配置し, practice で実行プログラムのあるディレクトリの read ディレクトリ内と自分のディレクトリ内の read ディレクトリ内のファイルを読み込めるようすると, 個人的な演習ファイルをサーバへ設置することが実現できる.
- practice 時に保存するファイル名を入力し, review 時に演習したいファイルを選択できる.

5.2.1 fork について

本プログラムは fork() 関数を使うことで複数の端末へ同時にサービスを提供できる. サーバプログラムを実行した際はプロセスが 1 つしか走っていないが, クライアントプログラムとつながると fork() するためプロセスが 2 つになる. 一方はクライアントにサービスを提供し, 他方は次のクライアントからの接続要求を待つ. また, クライアントとの接続が切れたプロセスは終了するようにしている. このようにしてサーバは常にクライアントの数より 1 つ多いプロセスが走行するようにしている.

プロセスを複数走行させる際, それぞれに異なるポートを占有させるようにしている. 4000 番を基準に bind できなければ 1 つ大きいポート番号の bind を試みるというループをし, bind できればループから離脱するというアルゴリズムを採用している. bind できた際にそのポート番号を出力するため, その番号でクライアントは connect を試みるとよい.

子プロセスは終了するとプロセスの機能はなくなるが, プロセステーブルには残ってしまう. 子プロセスの生成と終了を繰り返すとゾンビプロセスと呼ばれるこのプロセスが増加し, 一定数に達すると子プロセスが生成できなくなる. そのため, waitpid() システムコールでプロセステーブルから終了した子プロセスの内容を消去する手法を採用した.

6 実行結果に関する考察

本プログラムではコマンドを英字で入力することを要求するため, 問いが日本語で答えが英語という形式にしている. しかし, 問いと答えはカンマ区切りで分けられ, その前後で一方と他方を判別しているだけであり, 問いと答えを反対にしても正誤や問題等の送受信に影響は無いため演習は正常にできる. また, 問いと答えが日本語の問題なども実行できる.

また, 上記の通り問いと答えはカンマ区切りであり, 一行に問いと答えを収めるように要求しているため, 問題を細かく改行して表示したり, 長すぎる問題は使用できない. さらに, カンマを表示させたり複数に区切ることが実装できていない. よって以下のような仕様の変更案が考えられる.

- ,, の入力で区切り文字としてでなく出力用のカンマと認識するように変更する
  →問題にカンマが使用できるようになる.
- 波括弧などでくくられた文字列内を一つのまとまりと判断する
  →問題にカンマが使用できるほか, 複数の答えの保持などの実装も可能になる

- 区切り文字がくるまで 1 行受け取りを続ける
  →一行ごとに改行をさせることで問題に改行が実装できるほか, 送受信のデータ容量を気にすること無く問題の容量を増やすことができる
  →しかしこれは複数のデータを送るか一つのデータを送るかという判断が必要になるため, 送受信の手順を少し変更しなければならない

7 プログラムコード

クライアントプログラムとサーバプログラムのコードは以下である.

7.1 クライアントプログラム

```
 1  #include <sys/types.h>
 2  #include <sys/socket.h>
 3  #include <netdb.h>
 4  #include <unistd.h>
 5  #include <stdio.h>
 6  #include <stdlib.h>
 7  #include <string.h>
 8  #include <strings.h>
 9  #include <arpa/inet.h>
10
11  int fd;
12  char send_buf[1024];
13  char recv_buf[1024];
14  int stl;
15  int i=0;
16  int printflag=0;
17  int accountflag=0;
18  FILE *dfpc;
19  int rflag=0;
20
21  void check(int x){
22    printf("check %d\n",x);
23  }
24
25  void sendbuf(){
26    if(send(fd,send_buf,strlen(send_buf),0)==-1){
27      printf("error:send\n");
28      exit(0);
29    }
30    if(printflag==1){
31      fprintf(dfpc,"debugs %s\n",send_buf);
32      fprintf(dfpc,"================================\n");
33    }
34    if(printflag==2){
35      printf("=============send==================\n");
36      printf("%s\n",send_buf);
37      printf("=============send==================\n");
38    }
39    //bzero(send_buf,sizeof(send_buf));
40  }
41
42  void recvbuf(){
43    bzero(recv_buf,sizeof(recv_buf));
44    if(recv(fd,recv_buf,1000,0)==-1){
45      printf("error:recv\n");
46      exit(0);
47    }
48    if(printflag==1){
49      fprintf(dfpc,"debugr %s\n",recv_buf);
50      fprintf(dfpc,"********************************\n");
```

```
51    }
52    if(printflag==2){
53      printf("**********recv**********\n");
54      printf("%s\n",recv_buf);
55      printf("**********recv**********\n");
56    }
57  }
58
59
60  int subst(char *str, char c1, char c2){
61    int n = 0;
62    while(*str){
63      if(*str == c1){
64        *str = c2;
65        n++;
66      }
67      str++;
68    }
69    return n;
70  }
71
72  int send_file_data(char *buf){
73    FILE *fp;
74
75    subst(buf,'\n','\0');
76    if((fp = fopen(buf,"r")) == NULL){
77      fprintf(stderr,"ファイルがありません\n");
78      return 0;
79    }
80
81    sendbuf();
82    recvbuf();
83
84    while(fgets(send_buf,1000,fp)!=NULL){
85      sendbuf();
86      printf("%s\n",send_buf);
87      recvbuf();
88    }
89    bzero(send_buf,sizeof(send_buf));
90    sprintf(send_buf,"EOF");
91    sendbuf();
92    bzero(send_buf,sizeof(send_buf));
93    fclose(fp);
94    return 1;
95  }
96
97  int start_option(){                          // return 0 で成功
98    bzero(send_buf,sizeof(send_buf));
99    recvbuf();                        // buf = モードを選んでください
100   printf("%s\n",recv_buf);
101   read(0,send_buf,1000);
102   sendbuf();
103   if(strcmp(send_buf,"practice\n")==0) return 0;   // practice で break;
104   if(strcmp(send_buf,"quit\n")==0) return -1;
105
106   else if(accountflag==0){
107     if((strcmp(send_buf,"read\n")!=0)&&
108        (strcmp(send_buf,"new account\n")!=0)&&
109        (strcmp(send_buf,"log in\n")!=0)) return 1;
110   }else if(accountflag==1){
111     if(strcmp(send_buf,"review\n")==0){
112       rflag=1;
113       return 0;
114     }else if((strcmp(send_buf,"read\n")!=0)&&
115        (strcmp(send_buf,"log out\n")!=0)) return 1;
116   }
117   recvbuf();
118   if(strcmp(recv_buf,"追加するファイル名を入力してください")==0){ // read
119     printf("%s\n",recv_buf);
```

```
120       read(0,send_buf,1000);
121       if(send_file_data(send_buf)){    // s s
122         //      recvbuf();
123         return 1;
124       }else{
125         sprintf(send_buf,"can't open file");
126         sendbuf();
127         return 1;
128       }
129     }else if(strcmp(recv_buf,"作成するアカウント名を入力してください")==0){
130       printf("%s\n",recv_buf);
131       read(0,send_buf,1000);
132       sendbuf();
133       recvbuf();
134       printf("%s\n",recv_buf);
135       sprintf(send_buf,"na end");
136       sendbuf();
137       return 1;
138     }else if(strcmp(recv_buf,"アカウント名を入力してください")==0){
139       printf("%s\n",recv_buf);
140       read(0,send_buf,1000);
141       sendbuf();
142       recvbuf();
143       if(strstr(recv_buf,"ログイン")) accountflag=1;
144       printf("%s\n",recv_buf);
145       sprintf(send_buf,"login");
146       sendbuf();
147       return 1;
148     }else if(strcmp(recv_buf,"ログアウトしました")==0){
149       printf("%s\n",recv_buf);
150       accountflag=0;
151       sprintf(send_buf,"logout");
152       sendbuf();
153       return 1;
154     }
155     return 1;
156   }
157
158   int main(int argc, char *argv[]){
159     //int fd;
160     struct hostent *host;
161     struct sockaddr_in sa;
162     int pnum=0;
163     int i;
164     FILE *fp;
165     int eflag;
166
167     if(argc==2) pnum = atoi(argv[1]);
168     //printflag = atoi(argv[2]);
169
170     if((host = gethostbyname("localhost")) == NULL){
171       printf("error: gethostbyname\n");
172       return 1;
173     }
174
175     if((fd = socket(AF_INET, SOCK_STREAM, 0)) == -1){
176       printf("error: socket");
177       return 1;
178     }
179
180     sa.sin_family = host->h_addrtype;
181     bzero((char *)&sa.sin_addr, 12);
182     memcpy((char *)&sa.sin_addr, (char *)host->h_addr, host->h_length);
183     sa.sin_port = htons(pnum);
184
185     if(connect(fd,(struct sockaddr *)&sa, sizeof(sa))==-1){
186       printf("error:connect\n");
187       return 1;
188     }
```

```
189
190    dfpc = fopen("debugc.txt","w");
191
192    while(1){
193      while((eflag=start_option())==1);
194      if(eflag==-1)break;
195      /* 練習スタート */
196        recvbuf();
197      if(rflag==0){
198        while(1){
199          printf("%s\n",recv_buf);
200          stl = read(0,send_buf,1000);
201          sendbuf();
202          recvbuf();                                          // buf = "start\n"
203          if(strcmp(recv_buf,"start\n")==0) break;
204        }
205      }
206      printf("\n\n%s\n",recv_buf);
207      sprintf(send_buf,"recvstart\n");
208      sendbuf();
209      recvbuf();                                           // 1問目
210      while(strncmp(recv_buf,"result",6)){
211        printf("%s\n",recv_buf);
212        bzero(send_buf,sizeof(send_buf));
213        stl=read(0,send_buf,1000);
214        sendbuf();
215        recvbuf();
216        printf("%s\n",recv_buf);
217        sprintf(send_buf,"next\n");
218        sendbuf();
219        recvbuf();
220      }                                                   // recvでループを抜ける
221      printf("%s\n",recv_buf);
222      sprintf(send_buf,"practice end");
223      sendbuf();
224      rflag=0;
225    }
226    fclose(dfpc);
227    if(close(fd) == -1){
228      printf("error: close\n");
229      return 1;
230    }
231
232    return 1;
233  }
```

7.2 サーバプログラム

```
 1  #include <sys/fcntl.h>
 2  #include <sys/types.h>
 3  #include <sys/socket.h>
 4  #include <netinet/in.h>
 5  #include <netdb.h>
 6  #include <stdio.h>
 7  #include <stdlib.h>
 8  #include <string.h>
 9  #include <ctype.h>
10  #include <dirent.h>
11  #include <unistd.h>
12  #include <errno.h>
13  #include <sys/wait.h>
14  #include <signal.h>
15
16  #define DATA_MAX 1024
17  #define STR_LEN 1024
18
19  int data_count=0;
20
```

```
21  char send_buf[STR_LEN + 1];
22  char recv_buf[STR_LEN + 1];
23  int sockfd;
24  int new_sockfd;
25  int printflag=0;
26  FILE *dfps;
27  int accountflag=0;
28  char accountname[50];
29  int rflag=0;
30
31  void sendbuf(){
32    if(send(new_sockfd, send_buf, strlen(send_buf),0) == -1){
33      printf("error : send\n");
34      exit(0);
35    }
36    if(printflag==1){
37      fprintf(dfps,"debugs %s\n",send_buf);
38      fprintf(dfps,"================================\n");
39    }
40    if(printflag==2){
41      printf("=============send==================\n");
42      printf("%s\n",send_buf);
43      printf("=============send==================\n");
44    }
45  }
46  void recvbuf(){
47    bzero(recv_buf,sizeof(recv_buf));
48    if(recv(new_sockfd,recv_buf,1000,0) == -1){
49      printf("error : recv\n");
50      exit(0);
51    }
52    if(printflag==1){
53      fprintf(dfps,"debugr %s\n",recv_buf);
54      fprintf(dfps,"********************************\n");
55    }
56    if(printflag==2){
57      printf("************recv******************\n");
58      printf("%s\n",recv_buf);
59      printf("************recv******************\n");
60    }
61  }
62
63  void sendrecv(){
64    sendbuf();
65    recvbuf();
66  }
67
68  struct tango{
69    char japanese[STR_LEN];
70    char english[STR_LEN];
71  };
72
73  struct tango tango[DATA_MAX];
74
75  int subst(char *str, char c1, char c2){
76    int n = 0;
77    while(*str){
78      if(*str == c1){
79        *str = c2;
80        n++;
81      }
82      str++;
83    }
84    return n;
85  }
86
87  void dent(char *str){
88    subst(str,'\n','\0');
89  }
90
```

```
 91  void split(char *line,int i){
 92    int cnt=0;
 93
 94    for(cnt=0;*line != ',';cnt++){
 95      tango[i].japanese[cnt] = *line;
 96      line++;
 97    }
 98    line++;
 99    for(cnt=0;*line != '\n';cnt++){
100      tango[i].english[cnt] = *line;
101      line++;
102    }
103  }
104
105  int read_data(char filename[]){
106    FILE *fp;
107    char line[STR_LEN];
108    char str[20];
109
110    if(rflag==0){
111      sprintf(str,"./read/");
112      strcat(str,filename);
113    }
114    else sprintf(str,"./account/%s/mislog.csv",accountname);
115
116    if((fp = fopen(str,"r"))==NULL){
117      fprintf(stderr,"%s\n","error:can't read file.");
118      return 0;
119    }
120    for(data_count=0;(fgets(line, DATA_MAX + 1, fp) != NULL);data_count++)
121      split(line,data_count);
122
123    fclose(fp);
124
125    return 1;
126  }
127
128  void practice(){
129    int i;
130    int count=0;
131    FILE *fp;
132    char acfile[20];
133
134    if(accountflag==1){
135      sprintf(acfile,"./account/%s/mislog.csv",accountname);
136      if((fp=fopen(acfile,"w"))==NULL){
137        printf("error:practice mislog\n");
138      }
139    }
140
141    bzero(send_buf,sizeof(send_buf));
142    sprintf(send_buf,"start\n");
143    sendbuf();
144    recvbuf();                          //buf = "ok\n"
145    for(i=data_count;i>0;i--){
146      sprintf(send_buf,"%s(%d/%d)",
147              tango[data_count-i].japanese,
148              data_count-i+1, data_count);
149      sendbuf();
150      recvbuf();                      // ここで解答が送られる
151      printf("%s\n",recv_buf);
152      dent(recv_buf);
153      if(strcmp(recv_buf,tango[data_count-i].english)==0){
154        sprintf(send_buf,"correct\n\n");
155        count++;
156      }else{
157        sprintf(send_buf,"incorrect(%s)\n\n",tango[data_count-i].english);
158        if(accountflag==1)
159        fprintf(fp,"%s,%s\n",
160                tango[data_count-i].japanese,tango[data_count-i].english);
```

```
161       }
162       sendbuf();                        // 正誤判定を送信
163       recvbuf();                        // buf = "next\n"
164       printf("%s\n",recv_buf);
165     }
166     sprintf(send_buf,"result : %d / %d\n",count,data_count);
167     sendbuf();
168     recvbuf();
169     if(accountflag==1) fclose(fp);
170   }
171
172   int file_name(char *buf, char dirname[], char ext[]){
173     DIR *dirp;
174     struct dirent *p;
175     char str[1024];
176
177     bzero(buf,sizeof(buf));
178
179     if((dirp = opendir(dirname)) == NULL){
180       sprintf(buf,"Can't open directory %s\n",dirname);
181       return 1;
182     }
183     while((p = readdir(dirp)) != NULL){
184       if(strstr(p->d_name,ext)){
185         sprintf(str,"%s ",p->d_name);
186         strcat(buf,str);
187       }
188     }
189     strcat(buf,"\n");
190     if(closedir(dirp) != 0){
191       sprintf(buf,"Can't close directory %s\n",dirname);
192       return 1;
193     }
194     return 0;
195   }
196
197   int read_file(){
198     FILE *fp;
199     char file[50];
200
201     sprintf(file,"./read/");
202     sprintf(send_buf,"追加するファイル名を入力してください");
203     sendbuf();
204     recvbuf();
205     if(strcmp(recv_buf,"can't open file")==0) return 0;
206     dent(recv_buf);
207     strcat(file,recv_buf);
208     fp = fopen(file,"w");
209     sprintf(send_buf,"ok");
210     sendbuf();
211     // データ受取スタート
212     recvbuf();
213     while(strcmp(recv_buf,"EOF")!=0){
214       fprintf(fp,"%s",recv_buf);
215       sendbuf();
216       recvbuf();
217     }
218     fclose(fp);
219     return 0;
220   }
221
222   int new_account(){
223     int i;
224     char buf[100]="./account/";
225     sprintf(send_buf,"作成するアカウント名を入力してください");
226     sendbuf();
227     recvbuf();
228     dent(recv_buf);
229     strcat(buf,recv_buf);
```

```
230    i=mkdir(buf,0777);
231    if(i==0)sprintf(send_buf,"アカウントを作成しました");
232    else sprintf(send_buf,"このアカウントは使えません");
233    sendbuf();
234    recvbuf();
235    return i;
236  }
237
238  int log_in(){
239    DIR *dirp;
240    struct dirent *p;
241    int find=0;
242
243    sprintf(send_buf,"アカウント名を入力してください");
244    sendbuf();
245    recvbuf();
246    dent(recv_buf);
247    dirp = opendir("account");
248    while((p = readdir(dirp)) != NULL){
249      if(strcmp(p->d_name,recv_buf)==0) find=1;
250    }
251    if(find==0) sprintf(send_buf,"アカウント名が間違っています");
252    else{
253      sprintf(send_buf,"ログイン：%s",recv_buf);
254      accountflag=1;
255      strcpy(accountname,recv_buf);
256    }
257    sendbuf();
258    recvbuf();
259    return 0;
260  }
261
262  int log_out(){
263    accountflag=0;
264    bzero(accountname,sizeof(accountname));
265    sprintf(send_buf,"ログアウトしました");
266    sendbuf();
267    recvbuf();
268    return 0;
269  }
270
271  ////////////// 以下 main 部 ////////////////////
272
273  void check(int i){
274    printf("check%d\n",i);
275  }
276
277  void delete_child(){
278    while(waitpid(-1,NULL,WNOHANG)>0);
279    signal(SIGCHLD,delete_child);
280  }
281
282  int main(int argc,char *argv[]){
283    int writer_len;
284    struct sockaddr_in reader_addr;
285    struct sockaddr_in writer_addr;
286    char recv_msg[1024];
287    char send_msg[1024];
288    char buf[1024];
289    int i=0;
290    int pid;
291    pid_t ppid,cpid;
292    int pnum=4000;
293    int bindi=0;
294    int eflag=0;
295
296    //if(argc>1) pnum = atoi(argv[1]);
297    if(argc==2) printflag = atoi(argv[1]);
298
```

```
299     while(1){
300
301     /* ソケットの生成 */
302
303     if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
304       perror("reader: socket");
305       exit(1);
306     }
307
308     /* 通信ポート・アドレスの設定 */
309
310     bzero((char *) &reader_addr, sizeof(reader_addr));
311     reader_addr.sin_family = AF_INET;
312     reader_addr.sin_addr.s_addr = htonl(INADDR_ANY);
313     reader_addr.sin_port=htons(4000);
314
315       pnum=4000;
316       while(bind(sockfd,(struct sockaddr *)&reader_addr,sizeof(reader_addr))==-1){
317         pnum++;
318         reader_addr.sin_port=htons(pnum);
319       }
320       printf("pnum %d\n",pnum);
321
322       /* コネクト要求をいくつまで待つかを設定 */
323        if (listen(sockfd, 5) < 0) {
324         perror("reader: listen");
325         close(sockfd);
326         exit(1);
327       }
328        signal(SIGCHLD,delete_child);
329       /* コネクト要求を待つ */
330       writer_len = sizeof(struct sockaddr);
331       if ((new_sockfd = accept(sockfd,(struct sockaddr *)&writer_addr, &writer_len)) < 0) {
332         printf("reader: accept\n");
333         exit(1);
334       }
335       pid=fork();
336       if(pid==0)break;
337     }
338     close(sockfd);
339
340     dfps = fopen("debugs.txt","w");
341     accountflag=0;
342     bzero(accountname,sizeof(accountname));
343
344     while(1){
345       while(1){
346         if(accountflag==0){
347           sprintf(send_buf,"\n コマンドを入力してください\n");
348           strcat(send_buf,"practice\nread\nnew account\nlog in\nquit\n");
349           sendbuf();
350           recvbuf();
351           if(strcmp(recv_buf,"practice\n")==0)break;
352           else if(strcmp(recv_buf,"read\n")==0)read_file();
353           else if(strcmp(recv_buf,"new account\n")==0)new_account();
354           else if(strcmp(recv_buf,"log in\n")==0)log_in();
355           else if(strcmp(recv_buf,"quit\n")==0){
356             eflag=1;
357             break;
358           }
359         }else{
360           sprintf(send_buf,"\n コマンドを入力してください\n");
361           strcat(send_buf,"practice\nread\nreview\nlog out\nquit\n");
362           sendbuf();
363           recvbuf();
364           if(strcmp(recv_buf,"practice\n")==0)break;
365           else if(strcmp(recv_buf,"read\n")==0)read_file();
366           else if(strcmp(recv_buf,"log out\n")==0)log_out();
```

```
367           else if(strcmp(recv_buf,"review\n")==0){
368             rflag=1;
369             break;
370           }else if(strcmp(recv_buf,"quit\n")==0){
371             eflag=1;
372             break;
373           }
374         }
375       }
376       if(eflag)break;
377       if(rflag==0){
378         bzero(send_buf,sizeof(send_buf));
379         sprintf(send_buf,"\n 練習ファイル一覧\n");
380         if(file_name(buf,"./read",".csv")) exit(1);
381         sprintf(send_buf,"%s%s",send_buf,buf);
382         printf("%s\n",buf);
383         sprintf(buf,"\n ファイル名を入力してください");
384         strcat(send_buf,buf);
385         sendbuf();
386         recvbuf();
387         dent(recv_buf);
388       }
389       if(read_data(recv_buf))practice();
390       bzero(tango,sizeof(tango));
391       rflag=0;
392     }
393     sendbuf();
394     close(new_sockfd);
395     fclose(dfps);
396   }
```