プログラミング演習２（山崎班）

最終報告書

提出日　2018年6月26日

CY16173 八木橋晃一

**１．はじめに**

今回プロジェクト演習２の山崎班の最終課題において通信アプリコース２の「複数のクライアントが同時接続しても対応できるようなサーバーを作成する」を応用し、チャットのプログラムを作成したので本レポートにて報告する。本レポートでは、はじめに組み込み開発環境について述べたあとに、動作の概要、システムの詳細の順にのべ、最後にこの演習で習得した知識や技術、感想を述べる。

**２組み込み開発環境の概要**

**２.1演習全体を通しての組み込み開発のための機器、ソフトウェアとその役割について**

ハードウェアはDELL Vostro、Armadilloの２種類であり、ソフトウェアはLinux、 vi、gedit、minicomの3種類である。それぞれの概要と詳細を以下に示す。

DELL VostroはArmadillo開発におけるホストの役割をする。また開発に使用す るPCの名称である。

･OS:Linux(Debian)

･使用したPC：enshu-608-7

･ターゲットとはシリアルケーブルにて接続

･ターゲットのtftpブートのためLANに接続

･外付けHDDよりブート

･IP:172.28.34.107

･最終課題にも使用

Armadilloは開発の際に使用したターゲットである。

･OS:Linux

･使用したArmadillo：armadillo-7

･ホストとはシリアルケーブルにて接続

･tftpブートのためLANに接続

･IP:172.28.34.107

LinuxはホストとターゲットのOSである。

･ホストの「端末」にて操作

･最終課題にも使用

ｖi：テキストエディタ

･最終課題にも使用

gedit：テキストエディタ

･最終課題にも使用

minicom：ホストからシリアル通信でターゲットを制御すのに使用

･使用の際はホストの「端末」を開き、コマンドmcにて使用

**２.2最終課題における組み込み開発のための機器、ソフトウェアとその役割について**

ハードウェアはOptiPlex（大学のPC）を使用した。またソフトウェアはプログラムの編集に秀丸を使用した。

OptiPlex

･OS：Windows 7 Enterprise

秀丸エディタ：テキストエディタ

**２.3組み込みシステム開発の手順**

組み込み機器とは、ソフトウェアを用いて制御される機器の総称である。組み込み機器の大きな特徴としては、組み込み機器は何かの機能に特化しているものであるため、キーボードなどはついていないことと、処理能力が低いことの2つがあげられる。そのため、組み込み機器上で開発は行わず、組み込み機器をターゲットとし、ホストを別途用意する必要がある。ホスト上でプログラムを作成、コンパイル、実行可能ファイルの作成まで行い、それをターゲットに送り、実行させる。この点が普通の開発環境と異なる。プロジェクト演習２の山崎班ではDELL Vostroをホスト、 Armadilloをターゲットとして開発を行った。開発の手順としては、DELL Vostro内でプログラムの作成したのちにクロスコンパイルを行い、実行可能ファイルをコマンドscpにてArmadilloに転送し、実行させた。

**３．動作の概要**

本章では、最終課題で作成したプログラムについて説明する。作成したプログラムは通信アプリコースの２である「２つ以上のソケットの同時処理」を応用し、チャット機能を持たせたものである。このプログラムはサーバー側のプログラムとクライアント側のプログラムの２つから構成されている。それぞれの動作の概要と、チャットとして全体で見たときの動作の概要を以下に示す。

**3.1サーバー側のプログラムの動作の概要**

　　　サーバーのプログラムは、「＾ｃ」が入力されるか、特権ユーザーのクライアントからサーバーの終了コマンドである「#end」を受け取るまで処理を続ける。処理の内容としては新規の接続要求に対して、ユーザーの上限に達していない場合は接続を受け付け､すでに上限に達している場合は専用のソケットを用いてその旨を伝え、接続を切断する。名前が重複している場合はクライアント側に再度名前の入力を促す。本プログラムではユーザーの上限を２０人までとした。また、接続済みのクライアントからデータの受信があった場合、そのデータが登録されているコマンドに該当しないものは、送信者を除くクライアントに送信者の名前とデータを送信する。該当するコマンドが存在する場合は、そのコマンドの処理を行う。コマンドを以下に示す。

･$members：参加者一覧を表示

･$kick[名前]：対象者をキックする

･$re：最後に送信したものを再送信

･$stamp1：頭を抱えるスタンプを全員に送信

･$stamp2：ジョジョスタンプ1を全員に送信

･$stamp3：ジョジョスタンプ2を全員に送信

･$stamp4：ジョジョスタンプ3を全員に送信

･$[名前]@文またはスタンプ：対象者へプライベートチャット送信

･$logout：ログアウトする

･$pass:[パスワード]:特権ユーザーへ切り替える

･#end：サーバーを終了させる（特権ユーザーのみ）

**3.2クライアント側のプログラムの動作の概要**

　　クライアントのプログラムは接続要求をサーバーのプログラムに送り、設定したい名前を入力する。ユーザ－の数に空きがあり、名前に重複がみられない場合は設定した名前でチャットを開始する。空きがない場合は名前の入力をするとともに、その旨を伝えるメッセージが届き接続が切断される。名前の重複が見られる場合は再度名前の入力が促される。また、最後に送信した文章またはコマンドの保存、ログアウトとキックの処理、スタンプのデータ保管はクライアント側で行っている。そのため、サーバー側と同様に、入力したデータまたは受け取ったデータに該当するコマンド存在しない場合のみ、通常のメッセージとして送受信している。

**3.3全体の動作の概要**

　　　このチャットプログラムはソケット通信を利用してデータを送受信しているため、サーバー側とクライアント側が利用するLANが同一であれば、利用可能である。ただしクライアント側のIPアドレスの設定を対応させる必要がある。また、サーバーとクライアントという関係上、サーバー側を先に実行する必要がある。サーバー側は接続要求を受け付けられる状態に入り、クライアント側から接続要求があった場合、それを受け付け、接続を維持するか切断するかを判断する。データのやりとりは全てサーバーを経由し、サーバーはそのデータに該当するコマンドがあれば実行し、必要があればコマンドに対応したデータをクライアントに送信する。該当するコマンドがない場合は送り主の名前とともにデータをクライアントに送信する。クライアントは該当するコマンドがある場合はそれに対応する処理を行う。該当するコマンドが存在しない場合は、受信したデータをprintfで画面に出力する。

**４．システムの詳細**

本章では今回作成したプログラムの詳細について説明する。まず、サーバー側、クライアント側、全体のテストの実施例を列挙し、その後工夫した点に触れ、最後に作成したプログラムを載せている。

**4.1サーバー側のテスト**

･上限を３人に設定して、4人を接続させようとする

　３人目が接続した段階で、ユーザーの数が上限に達しことが伝えられ、4人目は名前を入力するとともに、ユーザーの数がすでに上限に達していることが伝えられ、切断された

**4.2クライアント側のテスト**

･コマンド$all$を入力

コマンド一覧が表示された

**4.3全体のテスト**

　　　･コマンド$stamp1～4を入力

それぞれに対応したスタンプが使用中のユーザーに表示された

　　　･コマンド$logoutを入力した

正常にログアウトができた

　　　･コマンド$membersを入力した

その時点で接続していたメンバーの一覧の名前が確認できた

　　　･コマンド$kickbbを入力した（ユーザーbbは接続している）

bbを正常にキックできた

･コマンド$kickで自分自身をキック使用とした

　　　　キック出来ずにログアウトを勧められた

･コマンド$kickで存在しないユーザーをキックしようした

　　　　メンバー一覧を参照するように勧められた

･最初の入力でコマンド$reを入力

　　　　$reの説明が表示された

･ああああを送信後コマンド$reを入力した

ああああが再送信された

　　　･コマンド$stamp1を送信後コマンド$reを入力した

　　　　stamp1再送信された

･コマンド$membersを送信後コマンド$reを入力した

メンバーの一覧が再び表示された

･コマンド$bbaaaaを入力した（ユーザーbbは接続している）

ユーザーbbにのみaaaaが送信された

･自分自身にプライベートチャットを送信しようとした

　プライベートチャットの説明が表示された

･存在しないユーザーに･自分自身にプライベートチャットを送信しようとした

　メンバーの一覧の参照を勧められた

･コマンドpass:mikuを入力した

　特権ユーザーになれた

･特権ユーザーでコマンドpass:mikuを入力した

　すでに特権ユーザーであることが表示された

･pass:rootを入力した

　パスワードの間違いを指摘された

･特権ユーザーでコマンド#endを入力した

　全てのユーザーの接続が切断され、サーバーのプログラムが終了した

･普通のユーザーでコマンド#endを入力した

　権限がない旨が表示された

･重複した名前で入ろうとした

　３度連続で試したが入室したとは表示されず、重複しない名前で入室を試みると

　入室したと表示された

･入室後にこんにちわと入力

　入室しているメンバーに送り主とこんにちわが表示された

･存在しないコマンド$12を入力

コマンド一覧を参照するように勧められた

**4.4工夫した点**

　　コマンドendを作成し、endと入力しパスワードを入力するとシャットダウンするようにした。

**4.5サーバー側のプログラム**

#include <arpa/inet.h>

#include<sys/socket.h>

#include<stdio.h>

#include<unistd.h>

#include<string.h>

#include<stdlib.h>

#include<sys/select.h>

int main(){

struct cliant{

int sockfd;

char name[20];

int super; //特権ユーザーは1

int situation; //-1のとき空いてるとする

//0の時は名前重複のため準備中

};

struct cliant user[20];

struct sockaddr\_in serv\_addr;

int i,s,t,l,bu,ki,pri; //カウンタ変数

int connect=0; //クライアントの接続数

int sockfd;

int maxfd=0; //selectの際の管理番号の最大値

int user\_num;

int pri\_flag; //プライベートチャットを送ろうとしたかのフラッグ

int out\_sockfd; //ユーザー数上限の旨を入れない人に伝えるため

char kickname[20];

char buff[100];

char back\_name[20];

char pri\_name[20];

char pri\_buff[121];

char co\_buff[121];

fd\_set readfds;

//空いているアカウントはsituation=-1

//全てを空いている状態に（初期化）

for(i=0;i<20;i++){

user[i].situation=-1;

}

//全てを一般ユーザーに（初期化）

for(i=0;i<20;i++){

user[i].super=0;

}

//ソケットを作る

sockfd=socket(AF\_INET,SOCK\_STREAM,0);

if(sockfd==-1){

perror("error\_serv\_socket");

exit(1);

}

//アドレスを作る

memset(&serv\_addr,0,sizeof(struct sockaddr\_in));

serv\_addr.sin\_family=AF\_INET;

serv\_addr.sin\_addr.s\_addr=htonl(INADDR\_ANY);

serv\_addr.sin\_port=htons(10000);

/\*

//デバック時のみ有効

//ソケットの再利用を許可

int flag=1;

if(setsockopt(sockfd,SOL\_SOCKET,SO\_REUSEADDR,&flag,sizeof(int))==-1){

perror("error\_serv\_setso");

exit(1);

}

\*/

//ソケットにアドレスを割り当てる

if(bind(sockfd,(struct sockaddr \*)&serv\_addr,sizeof(struct sockaddr\_in))==-1){

perror("error\_serv\_bind");

exit(1);

}

//コネクション要求を待ち始めるように指示

if(listen(sockfd,5)==-1){

perror("error\_serv\_listen");

exit(1);

}

while(1){

FD\_ZERO(&readfds); //FD\_SETの初期化

FD\_SET(sockfd,&readfds); //sockfd(接続要求があるか)を監視対象に追加

if(maxfd<sockfd) maxfd=sockfd; //最大値を更新

for(i=0;i<20;i++){

if((user[i].situation==1)||(user[i].situation==0)){ //使用中または準備中なら

FD\_SET(user[i].sockfd,&readfds); //select()で待つ読み込みソケットとして登録

if(user[i].sockfd>maxfd) maxfd=user[i].sockfd; //最大値を更新

}

}

if(select(maxfd+1,&readfds,NULL,NULL,NULL)==-1){

perror("error\_serv\_select");

exit(1);

}

if((FD\_ISSET(sockfd,&readfds)==1)&&(connect<20)){ //接続要求があった場合かつまだ空きがある場合

connect++;

for(i=0;i<20;i++){

if(user[i].situation==-1){ //未使用

user[i].sockfd=accept(sockfd,NULL,NULL); //一番若い未使用に割り当てる

if(user[i].sockfd==-1){ //acccept()のエラーチェック

perror("error\_serv\_accept");

exit(1);

}

if(read(user[i].sockfd,back\_name,20)==-1){ //送られてきた名前のバックアップをとる

perror("error\_serv\_read\_backname"); //重複しないかの判断のため

exit(1);

}

for(s=0;s<20;s++){

if((strcmp(user[s].name,back\_name)==0)&&(user[s].situation==1)){ //使用中のアカウントと名前が重複

if(write(user[i].sockfd,"NOT",10)==-1){ //名前の重複を伝える

perror("error\_serv\_write\_name\_errorsend");

exit(1);

}

user[i].situation=0; //準備中に設定

//内容のやりとりはできないが未使用と判断されて上書きされないように

break; //同じ名前が一人でもいたら

}

}

if(s==20){ //最後までみても名前の重複が確認されないとき

strcpy(user[i].name,back\_name); //バックアップの名前を割り当てる

if(write(user[i].sockfd,"OK",10)==-1){ //名前が重複しなかったことを伝える

perror("error\_serv\_write\_name\_OKsend");

exit(1);

}

}

else break; //名前が重複したので入室などの処理を行わない

user[i].situation=1; //使用中に設定

if(sprintf(co\_buff,"%sさんが入室しました",user[i].name)==-1){

perror("error\_serv\_sprintf\_enter");

exit(1);

}

for(l=0;l<20;l++){

if(user[l].situation==1){ //使用中の人に

if(write(user[l].sockfd,co\_buff,121)==-1){ //入室があったことを伝える

perror("error\_serv\_write\_enter");

exit(1);

}

}

}

if(connect==20){ //ユーザーの上限に達した

for(i=0;i<20;i++){

if(user[i].situation==1){

if(write(user[i].sockfd,"ユーザーの数が上限に達しました",121)==-1){ //上限を使用者に伝える

perror("error\_serv\_write\_max\_send");

exit(1);

}

}

}

}

break;

}

}

}

else if((connect==3)&&(FD\_ISSET(sockfd,&readfds)==1)){ //上限にも関わらず、接続要求があったとき

out\_sockfd=accept(sockfd,NULL,NULL); //ユーザーが上限に達していることを伝えるため繋ぐ

if(write(out\_sockfd,"OK",10)==-1){ //クライアント側の名前の重複判定のプロセスを飛ばさせるため

perror("error\_serv\_write\_processbreak\_send");

exit(1);

}

if(write(out\_sockfd,"既にユーザーの数が上限に達しています",121)==-1){ //入れないことを伝える

perror("error\_serv\_write\_maxed\_send");

exit(1);

}

if(write(out\_sockfd,"$kick",121)==-1){ //KICKを伝える(上限のため)

perror("error\_serv\_write\_kick\_send\_tobemax");

exit(1);

}

sleep(1);

if(close(out\_sockfd)==-1){ //通達用のソケットを閉じる

perror("error\_serv\_maxed\_close");

exit(1);

}

}

for(t=0;t<20;t++){

if((user[t].situation==1)&&(FD\_ISSET(user[t].sockfd,&readfds)==1)){ //使用中のアカウントから送られてきた場合

if(read(user[t].sockfd,buff,100)==-1){ //クライアントからメッセージを受け取る

perror("error\_serv\_read\_message\_receive");

exit(1);

}

if(strcmp(buff,"$logout")==0){ //送られた内容が$logoutコマンドだったら

sleep(1);

if(close(user[t].sockfd)==-1){ //対象者とのソケットを閉じる

perror("error\_serv\_close\_logout");

exit(1);

}

user[t].situation=-1;

user[t].super=0;

connect--; //使用者のカウントを減らす

if(sprintf(co\_buff,"%sさんが退出しました",user[t].name)==-1){

perror("error\_serv\_sprintf\_logout");

exit(1);

}

for(i=0;i<20;i++){ //使用中の全員に

if(user[i].situation==1){

if(write(user[i].sockfd,co\_buff,121)==-1){ //退出メッセージを送る

perror("error\_serv\_write\_logoutmessage\_send");

exit(1);

}

}

}

}

else if(strcmp(buff,"$members")==0){ //メンバー一覧を表示させるコマンドのとき

for(i=0;i<20;i++){

if(user[i].situation==1){

if(write(user[t].sockfd,user[i].name,121)==-1){ //メンバー一覧を伝える

perror("error\_serv\_write\_kick\_send");

exit(1);

}

}

}

}

else if(strncmp(buff,"$pass:",6)==0){ //パスワードの入力を行おうとした

if(strcmp(buff,"$pass:miku")==0){ //正しいパスワードが入力されたとき

if(user[t].super==0){

user[t].super=1;

if(write(user[t].sockfd,"特権ユーザーに切り替えました",121)==-1){ //特権ユーザーになったことを伝える

perror("error\_serv\_write\_besuper\_send");

exit(1);

}

}

else if(user[t].super==1){ //すでに特権ユーザーのとき

if(write(user[t].sockfd,"既に特権ユーザーです",121)==-1){ //すでに特権ユーザーであることを伝える

perror("error\_serv\_write\_already\_super\_send");

exit(1);

}

}

}

else{ //パスワードが異なる時

if(write(user[t].sockfd,"パスワードが違います",121)==-1){ //パスワードが違うことを伝える

perror("error\_serv\_write\_pass\_erroer\_send");

exit(1);

}

}

}

else if(strcmp(buff,"#end")==0){ //サーバー終了コマンド

if(user[t].super==1){ //コマンド入力者が特権ユーザーなら

for(i=0;i<20;i++){

if((user[i].situation==1)||(user[i].situation==0)){ //使用中または準備中のユーザーに

if(write(user[i].sockfd,"特権ユーザによりサーバーが停止しました",121)==-1){ //サーバーの停止を伝える

perror("error\_serv\_write\_endmessage\_send");

exit(1);

}

if(write(user[i].sockfd,"$kick",121)==-1){ //全てのユーザーをキックする

perror("error\_serv\_write\_endkick\_send");

exit(1);

}

sleep(1);

}

}

if(close(sockfd)==-1){ //listenのソケットを閉じる

perror("error\_serv\_close\_listen");

exit(1);

}

return 0;

}

else{

if(write(user[t].sockfd,"権限がありません",121)==-1){ //一般ユーザーにアクセス権限がないことを伝える

perror("error\_serv\_write\_notsuper\_send");

exit(1);

}

}

}

else if((strcmp(buff,"$stamp1")==0)||(strcmp(buff,"$stamp2")==0)||(strcmp(buff,"$stamp3")==0)||(strcmp(buff,"$stamp4")==0)){

//何かしらのスタンプコマンドが送られてきたら

sprintf(co\_buff,"%s:",user[t].name);

printf("co\_buff=%s",co\_buff);

for(i=0;i<20;i++){

if((i!=t)&&(user[i].situation==1)){

if(write(user[i].sockfd,co\_buff,121)==-1){ //送り主を伝える

perror("error\_serv\_write\_STAMP\_namesend");

exit(1);

}

}

if(user[i].situation==1){

if(write(user[i].sockfd,buff,121)==-1){ //スタンプコマンドを送る

perror("error\_serv\_write\_STAMP\_send");

exit(1);

}

}

}

}

else if(strncmp(buff,"$kick",5)==0){ //キックコマンドが送られてきたら

ki=0;

for(bu=5;bu<25;bu++){//キック対象者の名前を取得

kickname[ki]=buff[bu];

ki++;

if(buff[bu]=='\0') break;

}

}

for(i=0;i<20;i++){

//名前一致かつ使用中かつ送り主じゃない（未使用にしても名前は消していないので）かつ特権ユーザーではない

if((strcmp(user[i].name,kickname)==0)&&(user[i].situation==1)&&(strcmp(user[t].name,kickname)!=0)&&(user[i].super==0)){

if(write(user[i].sockfd,"$kick",121)==-1){ //KICKを伝える

perror("error\_serv\_write\_kick\_send");

exit(1);

}

sleep(1);

if(close(user[i].sockfd)==-1){

perror("error\_serv\_kick\_close");

exit(1);

}

user[i].situation=-1;

connect--;

if(sprintf(co\_buff,"%sさんが%sさんをキックしました",user[t].name,user[i].name)==-1){

perror("error\_serv\_sprintf\_kick");

exit(1);

}

for(s=0;s<20;s++){

if(user[s].situation==1){ //使用者全員に

if(write(user[s].sockfd,co\_buff,121)==-1){ //誰が誰をキックしたのかを伝える

perror("error\_serv\_write\_message\_send");

exit(1);

}

}

}

break;

}

else if(strcmp(user[t].name,kickname)==0){ //自分自身をキックしようとしたとき

if(write(user[t].sockfd,"自分自身はキックできません",121)==-1){

perror("error\_serv\_write\_message\_kickself");

exit(1);

}

if(write(user[t].sockfd,"退出のコマンドは$logoutです",121)==-1){ //ログアウトを勧める

perror("error\_serv\_write\_message\_kickadvice");

exit(1);

}

break;

}

else if(i==19){ //キックの対象者が存在しないとき

if(write(user[t].sockfd,"キック対象者が存在しません",121)==-1){

perror("error\_serv\_write\_message\_kicknone");

exit(1);

}

if(write(user[t].sockfd,"メンバーの一覧はコマンド$membersでみられます",121)==-1){ //メンバーの確認を勧める

perror("error\_serv\_write\_message\_kicknone\_advice");

exit(1);

}

}

}

}

else if(strncmp(buff,"$",1)==0){

pri=0;

pri\_flag=0;

for(i=1;i<22;i++){ //名前が20バイト＋@

if(buff[i]=='@'){ //相手の名前を取得

pri\_flag=1;

break;

}

pri\_name[pri]=buff[i];

pri++;

}

pri\_name[pri]='\0';

i++;

for(s=0;s<20;s++){

if((strcmp(user[s].name,pri\_name)==0)&&(user[s].situation==1)&&(strcmp(user[t].name,pri\_name)!=0)){

for(l=0;l<120;l++){

pri\_buff[l]=buff[i];

if(buff[i]=='\0')

break;

i++;

}

//printf("pri\_buff=%s\n",pri\_buff);

sprintf(co\_buff,"%s>>",user[t].name); //送り先でコマンドの認識が出きるように

//名前と内容は別々に送信

if(write(user[s].sockfd,co\_buff,121)==-1){ //送り主を伝える

perror("error\_serv\_write\_priname\_send");

exit(1);

}

if(write(user[s].sockfd,pri\_buff,121)==-1){ //送られてきたメッセージを送る

perror("error\_serv\_write\_primessage\_send");

exit(1);

}

break;

}

else if(strcmp(user[t].name,pri\_name)==0){ //自分自身にプライベートチャット

if(write(user[t].sockfd,"自分自身とチャットはできません",121)==-1){

perror("error\_serv\_write\_primessage\_sendself");

exit(1);

}

if(write(user[t].sockfd,"$名前@〜はプライベートチャットのコマンドです",121)==-1){ //プライベートチャットの説明

perror("error\_serv\_write\_primessage\_advice");

exit(1);

}

break;

}

else if((s==19)&&(pri\_flag==1)){

if(write(user[t].sockfd,"チャット対象者が存在しません",121)==-1){ //チャット対象者がいない

perror("error\_serv\_write\_message\_none");

exit(1);

}

if(write(user[t].sockfd,"メンバーの一覧はコマンド$membersでみられます",121)==-1){ //メンバーの確認を勧める

perror("error\_serv\_write\_primessage\_noneadvice");

exit(1);

}

}

else if((s==19)&&(pri\_flag==0)){

if(write(user[t].sockfd,"$はコマンドに使用します",121)==-1){ //$で始まっているがどのコマンドにも当てはまらない

perror("error\_serv\_write\_message\_none$");

exit(1);

}

if(write(user[t].sockfd,"コマンド一覧はコマンド$all$でみられます",121)==-1){ //コマンドの確認を勧める

perror("error\_serv\_write\_message\_$all$\_advice");

exit(1);

}

}

}

}

else { //通常のメッセージ

sprintf(co\_buff,"%s:%s",user[t].name,buff); //cliのprintfの\nと被るので、\n不要

//printf("co\_buff=%s\n",co\_buff);

for(i=0;i<20;i++){

if((i!=t)&&(user[i].situation==1)){ //送り主以外に

if(write(user[i].sockfd,co\_buff,121)==-1){ //送られてきたメッセージを送る

perror("error\_serv\_write\_message\_send");

exit(1);

}

}

}

}

}

if((user[t].situation==0)&&(FD\_ISSET(user[t].sockfd,&readfds)==1)){ //準備中のアカウントから送られてきた場合

//名前を受けとってもまだ入力があると思われないように、使用中、準備中の順でみてる

if(read(user[t].sockfd,back\_name,20)==-1){ //送られてきた名前のバックアップをとる

perror("error\_serv\_read\_backname2");

exit(1);

}

for(s=0;s<20;s++){

if((strcmp(user[s].name,back\_name)==0)&&(user[s].situation==1)){

printf("test1\n");

if(write(user[t].sockfd,"NOT",10)==-1){ //名前の重複を伝える

perror("error\_serv\_write\_name\_errorsend");

exit(1);

}

break; //同じ名前が一人でもいたら

}

}

if(s==20){ //最後までみても名前の重複が確認されないとき

strcpy(user[t].name,back\_name); //バックアップの名前を割り当てる

if(write(user[t].sockfd,"OK",10)==-1){ //名前が重複しなかったことを伝える

perror("error\_serv\_write\_name\_OKsend2");

exit(1);

}

user[t].situation=1; //使用中に設定

sprintf(co\_buff,"%sさんが入室しました",user[t].name);

for(l=0;l<20;l++){

if(user[l].situation==1){ //使用中の人に

if(write(user[l].sockfd,co\_buff,121)==-1){

perror("error\_serv\_write\_enter2"); //入室を全員に伝える

exit(1);

}

}

}

if(connect==20){ //ユーザーの上限に達した

for(i=0;i<20;i++){

if(user[i].situation==1){

if(write(user[i].sockfd,"ユーザーの数が上限に達しました",121)==-1){ //上限を使用者に伝える

perror("error\_serv\_write\_max\_send2");

exit(1);

}

}

}

}

}

}

}

}

return 0;

}

**4.6クライアント側のプログラム**

#include<arpa/inet.h>

#include<sys/socket.h>

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<string.h>

#include<sys/select.h>

int main(){

struct sockaddr\_in serv\_addr;

int sockfd,p\_connect,p\_write,p\_close,p\_read;

int i;

int able=0; //初回の入力かを確認

fd\_set readfds;

char buff[121];

char back\_buff[121];

char name\_buff[20];

sockfd=socket(AF\_INET,SOCK\_STREAM,0);

if(sockfd==-1){

perror("error\_cil\_socket");

exit(1);

}

memset(&serv\_addr,0,sizeof(struct sockaddr\_in));

serv\_addr.sin\_family=AF\_INET;

serv\_addr.sin\_addr.s\_addr=inet\_addr("172.28.34.67");

serv\_addr.sin\_port=htons(10000);

printf("名前を入力してください\n");

scanf("%s",name\_buff); //servで他のクライアント割り当て待ちにならないようにconnect前

//コネクションを張るための要求をサーバーに送る

p\_connect=connect(sockfd,(struct sockaddr \*)&serv\_addr,sizeof(struct sockaddr\_in));

if(p\_connect==-1){

perror("error\_cil\_connect");

exit(1);

}

if(write(sockfd,name\_buff,20)==-1){

perror("error\_cil\_write\_namesend");

exit(1);

}

while(1){

FD\_ZERO(&readfds);

FD\_SET(sockfd,&readfds);

FD\_SET(0,&readfds);

if(select(sockfd+1,&readfds,NULL,NULL,NULL)==-1){

perror("error\_cil\_nameselect");

exit(1);

}

if(FD\_ISSET(sockfd,&readfds)==1){

if(read(sockfd,buff,10)==-1){ //名前が重複しているかの結果を受け取る

perror("error\_serv\_read\_resultname\_receive");

exit(1);

}

if(strcmp(buff,"OK")==0) break; //名前が重複していなかったのでチャットへ

else{ //名前が重複しているので、再度名前設定

printf("その名前はすでに使われています\n");

printf("再度名前を入力してください\n");

scanf("%s",name\_buff);

if(write(sockfd,name\_buff,20)==-1){

perror("error\_cil\_write\_renamesend");

exit(1);

}

}

}

}

while(1){

FD\_ZERO(&readfds);

FD\_SET(0,&readfds);

FD\_SET(sockfd,&readfds);

if(select(sockfd+1,&readfds,NULL,NULL,NULL)==-1){

perror("error\_cil\_select");

exit(1);

}

if(FD\_ISSET(0,&readfds)==1){ //キーボード入力がある場合

scanf("%s",buff);

if((strcmp(buff,"$re")==0)&&(able!=0)){ //最初の入力ではないかつコマンド$re

if(write(sockfd,back\_buff,100)==-1){ //前回と同じものを送信

perror("error\_cil\_write\_backsend");

exit(1);

}

if(strncmp(back\_buff,"$stamp",6)!=0){ //スタンプは自分にも送られてくるので除外

printf("%s\n",back\_buff); //再送信内容を自分にも表示

}

}

else if((strcmp(buff,"$re")==0)&&(able==0)){ //最初の入力でかつコマンド$re

printf("参照する入力がありません\n"); //前回と同じものを送信

printf("コマンド$reは直近のコマンドの再実行またはメッセージの再送信です\n");

}

else if((strcmp(buff,"$all$")==0)){ //コマンド一覧を表示する

printf("$members：参加者一覧を表示\n");

printf("$kick[名前]：対象者をキックする\n");

printf("$re：最後に送信したものを再送信\n");

printf("$stamp1：頭を抱えるスタンプ\n");

printf("$stamp2：ジョジョスタンプ1\n");

printf("$stamp3：ジョジョスタンプ2\n");

printf("$stamp4：ジョジョスタンプ3\n");

printf("$[名前]@文：対象者へプライベートチャット送信\n");

printf("$logout：ログアウトする\n");

printf("$pass:[パスワード]:特権ユーザーへ切り替える\n");

printf("#end：サーバーを終了させる（特権ユーザーのみ）\n");

}

else{

if(write(sockfd,buff,100)==-1){ //通常メッセージをサーバーへ送信

perror("error\_cil\_write\_send");

exit(1);

}

able=1;

}

if(strcmp(buff,"$re")!=0){ //2回目以降のコマンド$reで保存されるの防ぐため

strcpy(back\_buff,buff); //送信したもののバックアップをとる

}

if(strcmp(buff,"$logout")==0){

printf("%sさんが退出しました\n",name\_buff);

break;

}

}

if(FD\_ISSET(sockfd,&readfds)==1){ //メッセージが着た場合

if(read(sockfd,buff,121)==-1){

perror("error\_cil\_read\_receive");

exit(1);

}

if(strcmp(buff,"$kick")==0){ //キックコマンドのとき

printf("キックされました\n");

break;

}

else if(strcmp(buff,"$stamp1")==0){

printf("　　 ＿\n");

printf(" ／∧ )∧ ⌒)\n");

printf("｜(Д｀/ /\n");

printf("｜レ、/ /＼\n");

printf("＼／(＿\_ノ＼\n");

printf("　　　/　　 ｜\n");

printf("　　 /　/＼　＼＿\n");

printf("　　(＿ヘ\_)＼＿\_ ヽ\n");

printf("　　　　　　 　(\_ノ\n");

}

else if(strcmp(buff,"$stamp2")==0){

printf("お　　　 ＿\_＿　　　 さ\n");

printf("れ　　 ／￣￣ ＼　 　 す\n");

printf("た　　// / |　 ∧ 　が\n");

printf("ち　 ｜＿＿＿＿\_｜\n");

printf("に (⌒ ｿ　　　　 `ﾚ⌒) デ\n");

printf("で　＼\_二二二二二\_／ ィ\n");

printf("き　 //=ヽノ≡ヽ( ヽ オ\n");

printf("な 〈((･ /　ｲ･)　Lノ ！\n");

printf("い　(| ~(ﾆヽ⌒　ﾉr)\n");

printf("事　 |　ﾄ┬┬ｲ ｜ｲ|\n");

printf("を＿ | ｜/￣~| ｜リ\n");

printf("／へ＼　ﾋL＿ノ ｜|\n");

printf("`へ ＼) `ー-′／ |\n");

printf("へ ＼ﾉ| ＼＿／ / |\n");

printf("Zヽﾉ(ノ ＼ ＼／　ヽ\n");

printf("￣\n");

}

else if(strcmp(buff,"$stamp3")==0){

printf(" 　　　　　 　へ\n");

printf("あそ平　 (￣￣￣￣￣ヽ\n");

printf("ここ然　( ヽ ヽ ノ　 ヽ\n");

printf("がにと　 >＜二二二＞､ノ\n");

printf("れシや　//へ　　ノ~ヽ|\n");

printf("るビっ (Y (･>ッ<･)　|)\n");

printf("！れて　|　-(＿)-　 Ｖ\n");

printf("　るの /|　r＝＝-、 /＼\n");

printf("　！け｜ヽ ＼/二ノ／/\n");

printf("　　る　＼＞——イ／\n");

printf("　　ッ　　ヽ　　 /\n");

}

else if(strcmp(buff,"$stamp4")==0){

printf("こいつ\n");

printf("最高に誇りの道を行く者\n");

printf("\n");

printf("　　　 \_\n");

printf("　　　( ヽ\n");

printf("　　　｜ |\n");

printf("　　　｜ |\n");

printf("　　　 ﾚ-|＿\_へ\n");

printf("　　　/ ／ ／￣＼\n");

printf("　　 ｜(ﾌ(~)(~)￣)､\n");

printf("　　 ()丿 /　/ イ |\n");

printf("　　 ｜| ｜ ｜　Lノ\n");

printf("　　 ｜　　　　｜\n");

printf("　　 ｜　　　　/\n");

printf("　　／￣￣￣/-<\n");

printf("　 //　 /　/　｜\n");

printf("　｜／~￣￣　 /\n");

printf("　/／￣￣￣＼/\n");

}

else{

printf("%s\n",buff);

}

}

}

//ソケットを終了する

if(close(sockfd)==-1){

perror("error\_cli\_close");

exit(1);

}

return 0;

}

**５．おわりに**

今回の演習で楽しそうだからというだけでチャットを選択したが、他の課題よりも結果が目に見えるので楽しかったです。また、selectでのキーボード入力の監視などはwrite、readの説明でさりげなくちゃんと0,1.2はそれぞれ何を意味するのかが触れられており、最初の授業で説明されていた通り、情報がうまくちりばめられていると感じました。今回のプログラムでは、時間がなくプライベートチャットでは自分にスタンプが表示されないこと、特定の単語を規制すること、１人を除いて送信すること、sockfdの値を出力するテスト、特権ユーザーをキックするテストおよびそれに対応したメッセージ、特権ユーザーによる乗っ取り、IPアドレスの変数化、ユーザー上限の定義化､エラーチェックや全体送信の関数化などが出来なかったことが残念でした。