情報科学演習C　課題4レポート

氏名　山久保孝亮
所属　大阪大学基礎工学部情報科学科ソフトウェア科学コース
メールアドレス　u327468b@ecs.osaka-u.ac.jp
学籍番号　09B22084

提出日　2024年8月4日
担当教員　平井健士, 中島悠太

1　課題 4-1

1.1　プログラムの仕様

課題 4-1 で作成したクライアントプログラム chatclient.c の仕様は以下の通りである.

- プログラム実行の書式は"./chatclient [サーバプログラムを実行中のホスト名] [使用したいユーザ名]"である.
- 接続できたかどうか，名前を登録できたかどうかが標準出力に表示される. 接続できた場合はチャット機能が使用でき, 接続できなかった場合はプログラムが終了する.
- チャット機能には以下の 3 つの機能がある.
  1. 標準入力に文字列を書き込んで [Enter] キーを押すと, サーバにその文字列を送信する.
  2. サーバに接続されているほかのホストから送られてきた文字列を受信し, 標準入力に表示させる.
  3. EOF を入力するとサーバとの接続が切れ, プログラムを終了する.

また, サーバプログラム chatserver.c の仕様は以下の通りである.

- プログラム実行の書式は引数はなしである. 即ち"./chatserver"である.
- 一度に接続できるユーザの数は 5 つである.
- 常に新たなユーザの接続を待っており, 新たに接続しようとするユーザが現れた場合には正常に接続できたかどうかと, すでに同じユーザ名が登録されていないかを確認する．どちらも満たしていればその旨の文字列を送信して新たなユーザとして登録し, いずれかを満たさない場合はその旨の文字列を送信して再び待ち状態に入る.
- 登録されたユーザから文字列を受信した場合は，その文字列の先頭にユーザ名を追加して送信してきたユーザ以外に追加した後の文字列を送信する.
- EOF を受信するとそのユーザの接続を切り, ユーザの情報を削除する.

また，両社のプログラムに共通する仕様は以下のとおりである.

- 使用するポート番号は 10140 である.
- ユーザ名は英数字，ハイフン，アンダースコアのみから成る.

1.2　クライアントプログラムのアルゴリズム

今回作成したプログラムは指導書の状態に則って作成しており, 以下の図 1 のフローチャートはそれぞれの状態の遷移の様子を表す.
それぞれの状態は指導書のとおりに実装した．また，図 1 には無いがそれぞれの状態の処理で異常が発生した場合は状態 6 として例外処理が存在する. それぞれの状態の概要は以下のとおりである.

状態 1: 初期状態
　ソケットを作成し，第一引数のホスト名に接続要求を出す.

状態 2: 参加
　サーバから接続できたかどうかの文字列を受け取る.

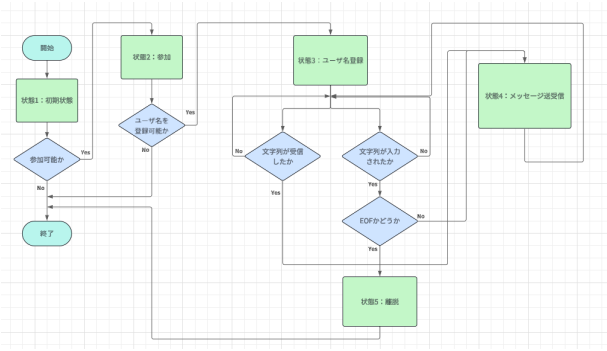

図 1: クライアントプログラムのフローチャート

状態 3: ユーザ名登録

第二引数のユーザ名を送信し，ユーザ名を登録できたかどうかの文字列を受け取る．

状態 4: メッセージ送受信

標準入力から文字列が入力されればその文字列をサーバへ送信する．サーバから文字列を受信すればその文字列を標準入力へ出力する．

状態 5: 離脱

標準入力が”EOF”のとき，ソケットを閉じてプログラムを正常終了する．

状態 6: 例外処理

サーバに接続拒否またはユーザ名の登録に失敗した際に，ソケットを閉じてプログラムを異常終了する．

1.3 クライアントプログラムの実装方法

ここではアルゴリズムで記述した各状態とその条件分岐の詳細な実装方法について記述する．以下の表 1 はこのプログラムで使用した変数名とその表す内容である．

変数名	型	表す内容
sock	int	ソケットディスクリプタ
n	int	入出力操作の結果を格納
rbuf	char	送受信する文字列を格納する．要素数は 1024
rfds	fd_set	ファイルディスクリプタのセット
tv	struct timeval	タイムアウト値を設定する構造体
*server	struct hostent	ホスト情報を格納するためのポインタ
svr	struct sockaddr_in	サーバのアドレス情報を格納する構造体
current_time	time_t	現在の時刻を格納
*local_time	struct tm	ローカルタイムを表す構造体へのポインタ
argc	int	引数の個数を格納
*argv[]	char	引数の文字列を格納

表 1: このプログラムで使用した変数

1.3.1 初期状態

状態 1 のプログラムは大きく以下のように処理が分かれる.

1. 書式の確認し, ソケットを作成
2. ホスト名を取得し，接続する

以下でその詳細について記述する.

1. この処理のプログラムは以下のようになる.

```
if(argc != 3){
    fprintf(stderr, "Usage: %s <hostname> <username>\n", argv[0]);
    exit(1);
}
if ((sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP))<0) {
    perror("socket");
    exit(1);
}
```

   コマンドライン引数が 3 であるかどうかを確認して書式が正しいかを確認する. ここで argc の値が 3 である理由は"./chatclient"も引数として数えられるためである．そして sock() を使ってソケットを作成する.

2. この処理のプログラムは以下のようになる.

```
server = gethostbyname(argv[1]);
if (server == NULL) {
    fprintf(stderr, "ERROR, no such host as %s\n", argv[1]);
    exit(0);
}
bzero((char *)&svr, sizeof(svr));
svr.sin_family = AF_INET;
bcopy((char *)server->h_addr, (char *)&svr.sin_addr.s_addr, server->h_length);
svr.sin_port = htons(10140);
if (connect(sock, (struct sockaddr *)&svr, sizeof(svr)) < 0) {
    perror("client: connect");
    close(sock);
    exit(1);
}
```

   コマンドライン引数の 1 番目の引数をホスト名として変数に格納する．ここでインデックスが 1 である理由は argv[0] に"./chatclient"が格納されているからである．そしてそのホスト名のサーバがあれば connect() を使って接続をサーバに要請する.

1.3.2 参加と参加できたかどうかの条件分岐

状態 2 のプログラムは以下のようになる.

```
bzero(rbuf, 1024);
n = read(sock, rbuf, 1024);
if(n <= 0){
    perror("Connection closed by client.\n");
```

```
5     close(sock);
6     return 0;
7 }else if(strcmp(rbuf,"REQUEST ACCEPTED\n") == 0) {状態３の処理
8
9 }else{状態６の処理
10 }
```

1 から 2 行目でサーバから文字列を受信する．正常に受信ができればその内容が”REQUEST ACCEPTED \n”と一致しているかどうかを確認し一致していれば状態 3 へ，一致していなければ状態 6 へ遷移する．文字列が一致しているかどうかの判定には strcmp() を使用した．

1.3.3 ユーザ名登録

状態３のプログラムは大きく以下のように処理が分かれる．

1. 接続成功の文字列を受け取り，ユーザ名をサーバへ送信する．
2. サーバからユーザ名の登録に関する文字列を受け取る．

以下でその詳細について記述する．

1. この部分の処理のプログラムは以下のようになる．

```
1  printf("%s",rbuf);
2  bzero(rbuf,1024);
3  strcpy(rbuf,argv[2]);
4  rbuf[strlen(argv[2])] = '\n';
5  n = write(sock, argv[2], strlen(argv[2]));
6  if(n < 0){
7      perror("ERROR writing");
8      close(sock);
9      return 0;
10 }
```

   まずサーバから受け取った文字列”REQUEST ACCEPTED\n”を標準入力に表示し，第二引数のユーザ名をサーバに送信する．引数を送信する際，ユーザ名の終わりを判別するために改行文字を最後に加える．また，rbuf を bzero() を使って初期化してからユーザ名を格納している．これをしていないと，例えば”aiueo”というユーザ名にした場合 rbuf に格納されている文字列が”aiueoST ACCEPED\n”という文字列になってしまうので前の文字列を初期化しなければならない．

2. この部分の処理のプログラムは以下のようになる．

```
1 bzero(rbuf, 1024);
2 n = read(sock, rbuf, 1024);
3 if(n <= 0){
4     perror("Connection closed by client.\n");
5     close(sock);
6     return 0;
7 }
```

   この処理でも bzero() で rbuf を初期化してから read() を使って文字列を受け取っている．

1.3.4 メッセージ送受信と待ち状態

状態４と待ち状態のプログラムは大きく以下のように処理が分かれる．

1. 状態３で最後に取得した文字列を標準入力に表示
2. 待ち状態によるソケットと標準入力の監視
3. 送信処理
4. 受信処理

以下でその詳細について記述する．

1. この処理のプログラムは以下のようになる．

```
printf("%s",rbuf);
```

2. この処理のプログラムは以下のようになる．

```
while(1){
    FD_ZERO(&rfds);
    FD_SET(0, &rfds);
    FD_SET(sock, &rfds);
    tv.tv_sec = 1;
    tv.tv_usec = 0;
    if (select(sock + 1, &rfds, NULL, NULL, &tv) > 0) {
        if (FD_ISSET(0, &rfds)) {送信

        }
        if (FD_ISSET(sock, &rfds)) {受信

        }
    }
}
```

while 文による無限ループの中で select() を使って標準入力とソケットを監視している．FD_ISET の第一引数が 0 であれば送信，sock であれば受信の処理へ移る．

3. この処理のプログラムは以下のようになる．

```
bzero(rbuf, 1024);
if (fgets(rbuf, 1024, stdin) == NULL) {状態５

}
n = write(sock, rbuf, strlen(rbuf));
if (n < 0) {
    perror("ERROR writing");
    break;
}
```

ここでは write() を使ってメッセージの送信の処理を実装している．fgets() によって標準入力から取得された文字列を rbuf へ格納して送信している．このとき，EOF が入力された時は状態５に遷移する．

4. この処理のプログラムは以下のようになる．

```
bzero(rbuf, 1024);
n = read(sock, rbuf, 1024);
if(n <= 0){
    perror("Connection closed by client.\n");
    close(sock);
    return 0;
}else{
    printf("%s",rbuf);
}
```

1.3.5 離脱

状態５のプログラムは以下のようになる．

```
printf("\nEOF detected\n");
close(sock);
exit(0);
```

EOF が入力されたことを表示し，ソケットを閉じて exit(0) でプログラムを正常終了する．

1.3.6 例外処理

状態 6 のプログラムは以下のようになる．

```
printf("%s",rbuf);
close(sock);
exit(1);
```

この状態はサーバ側から受け取った文字列が正常ではなかった場合の処理を実行するので，まずサーバから受け取った文字列を出力する．そしてソケットを閉じて exit(1) でプログラムを異常終了する．