ネットワークプログラミングII

総合演習

402　　411

2017年7月21日

1 概要

今回、ネットワークプログラミング II の総合演習として作成した作品は「六目並べ」である。

2 ソースコード

総合演習で作成したソースコードをリスト 1〜6 に示す。sessionman.h であるリスト 1 では、sessionman.c で使用するマクロや外部関数を宣言した。なお、このプログラムを実行するにはサーバの 50002 番ポートを開けておく必要がある。

リスト 1　sessionman.h

```
1  #include <stdlib.h>
2  #include <string.h>
3  #include <sys/types.h>
4  #include <netinet/in.h>
5
6  #define PORT    (in_port_t)50002
7  #define MAX_ATTENDANTS 5
8
9  extern void enter();
10 extern void sessionman_init(int num, int maxfd);
11 extern void sessionman_loop();
```

sessionman.c であるリスト 2 では、セッションに関する処理の定義をした。サーバーに接続した各クライアントには各ファイルディスクリプタ fd が割り当てられ、サーバとクライアントは fd を通して通信を行う。15 行目から 18 行目では各クライアントの情報を格納する構造体 ATTENDANT を定義した。25 行目の enter 関数では、i 番目のクライアントを割り当てられた fd で初期化する処理を行う。38 行目の sessionman_init 関数では、ゲームの先攻後攻を決める処理を行う。これはゲームが開始するときに呼び出される。63 行目の sessionman_loop 関数では、各クライアントから送られてきた内容を全てのクライアントに送信する処理を行う。89 行目の ending 関数では、サーバとクライアントの通信の終了処理を行う。101 行目の send_all 関数では、全てのクライアントにグローバル変数 buf の内容を送信する処理を行う。

リスト 2　sessionman.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <unistd.h>
5  #include <sys/types.h>
6
7  #define MAX_ATTENDANTS 5
8  #define BUF_LEN        80
9
10 static char buf[BUF_LEN];
11 static fd_set mask;
12 static int width;
13 static int attendants;
14
15 typedef struct {
16     int fd;
```

```
17      // char name[16];
18  } ATTENDANT;
19
20  static ATTENDANT p[MAX_ATTENDANTS];
21
22  static void send_all(int i, int n);
23  static void ending();
24
25  void enter(int i, int fd)
26  {
27      int len;
28      static char *mesg = "Wait.\n";
29
30      p[i].fd = fd;
31
32      // Send "Wait." to player who is first entered room.
33      if (i == 0) {
34          write(fd, mesg, strlen(mesg));
35      }
36  }
37
38  void sessionman_init(int num, int maxfd)
39  {
40      int i;
41      // static char *mesg = "Game Start.\n";
42      char message[20];
43      int rnd;
44
45      srandom(time(NULL));
46      rnd = random() % 2;
47
48      attendants = num;
49
50      width = maxfd + 1;
51      FD_ZERO(&mask);
52      FD_SET(0, &mask);
53      for (i = 0; i < num; i++) {
54          FD_SET(p[i].fd, &mask);
55      }
56
57      sprintf(message, ":%d Game Start.\n", rnd);
58      write(p[0].fd, message, strlen(message));
59      sprintf(message, ":%d Game Start.\n", 1 - rnd);
60      write(p[1].fd, message, strlen(message));
61  }
62
63  void sessionman_loop()
64  {
65      fd_set readOk;
66      int i;
67
68      while (1) {
69          readOk = mask;
70          select(width, (fd_set *)&readOk, NULL, NULL, NULL);
71
72          // Is there are input from keyboard?
73          if (FD_ISSET(0, &readOk)) {
74              ending();
```

```
75          }
76
77          for (i = 0; i < attendants; i++) {
78              if (FD_ISSET(p[i].fd, &readOk)) {
79                  int n;
80                  n = read(p[i].fd, buf, BUF_LEN);
81                  send_all(i, n);
82              }
83          }
84      }
85  }
86
87  // Sub routine
88
89  static void ending()
90  {
91      int i;
92      for (i = 0; i < attendants; i++) {
93          write(p[i].fd, "q", 1);
94      }
95      for (i = 0; i < attendants; i++) {
96          close(p[i].fd);
97      }
98      exit(0);
99  }
100
101 static void send_all(int i, int n)
102 {
103     int j;
104     for (j = 0; j < attendants; j++) {
105         write(p[j].fd, buf, n);
106     }
107 }
```

session.h であるリスト 3 では、session.c で使用するマクロや外部関数を宣言した。

リスト 3　session.h

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/types.h>
4  #include <netinet/in.h>
5
6  #define PORT (in_port_t)50002
7  #define HOSTNAME_LENGTH 64
8
9  extern void session_init(int soc);
10 extern void session_loop();
```

session.c であるリスト 4 では、六目並べのゲームの処理とサーバと通信を行う処理を行う。58 行目の session_init 関数ではソケットの初期化と、ゲームのために使用するライブラリ ncurses の初期化を行う。93 行目の session_loop 関数では、プレイヤーやサーバからの入力を監視するためのループを行う。109 行目の if 文以降では、標準入力から入力があった場合の処理を行う。標準入力からのキーの入力に対応する処理を表 1 に示す。148 行目の if 文以降では、サーバからデータが送られてきた場合の処理を行う。具体的に、送られてきた文字列とそれに対応する処理を表 2 に示す。

session.c であるリスト 4 のプライベート関数について、226 行目の init_goban 関数では、碁盤の初期化を行う。243 行目の is_my_turn 関数では、ゲームのステップ数と自分の石の文字を与えると、現在のターンが自分のターンであれば 1 を、相手のターンであれば 0 を返す。256 行目の put_stone 関数では、与えられた (x,y) 座標に石が何も置かれていない場合に、与えられた石の文字を碁盤に配置する。267 行目の die 関数では、ゲーム終了時の処理を行う。274 行目の detect_rokumoku 関数では、与えられた石の文字列が碁盤上で 6 つ以上ならんでいるかを調べる。石が 6 つ以上ならんでいれば 1 を、そうでなければ 0 を返す。

リスト 4　session.c

```
1  #include <stdlib.h>
2  #include <unistd.h>
3  #include <string.h>
4  #include <sys/types.h>
5  #include <signal.h>
6  #include <ncurses.h>
7
8  #define BUF_LEN 80
9
10 #define INFO_WIN_WIDTH  40
11 #define INFO_WIN_HEIGHT 1
12
13 #define GOBAN_SCREEN_HEIGHT 20
14 #define GOBAN_SCREEN_WIDTH  40
15
16 static char goban_my_stone;
17 static char goban_peer_stone;
18
19 static char goban_plane[GOBAN_SCREEN_HEIGHT][GOBAN_SCREEN_WIDTH] = {
20     ". . . . . .",
21     ". . . . . .",
22     ". . . . . .",
23     ". . . . . .",
24     ". . . . . .",
25     ". . . . . .",
26     ". . . . . .",
27     ". . . . . .",
28     ". . . . . .",
29     ". . . . . .",
30     ". . . . . .",
31     ". . . . . .",
32     ". . . . . .",
33     ". . . . . .",
34     ". . . . . .",
35     ". . . . . .",
36     ". . . . . .",
37     ". . . . . .",
38     ". . . . . .",
39     ". . . . . ."
40 };
41 static char goban_plane_orig[GOBAN_SCREEN_HEIGHT][GOBAN_SCREEN_WIDTH];
42
43 static WINDOW *win_info, *win_goban;
44 static WINDOW *frame_info, *frame_goban;
45
46 static char send_buf[BUF_LEN];
47 static char recv_buf[BUF_LEN];
48 static int session_soc;
```

```
49  static fd_set mask;
50  static int width;
51
52  static void init_goban();
53  static int is_my_turn(int, char);
54  static int put_stone(int, int, char);
55  static void die();
56  static int detect_rokumoku(char);
57
58  void session_init(int soc)
59  {
60      int i;
61      int x, y;
62      session_soc = soc;
63      width = soc + 1;
64      FD_ZERO(&mask);
65      FD_SET(0, &mask);
66      FD_SET(soc, &mask);
67
68      initscr();
69      signal(SIGINT, die);
70
71      win_info = newwin(INFO_WIN_HEIGHT, INFO_WIN_WIDTH, 22, 1);
72      scrollok(win_info, FALSE);
73      wmove(win_info, 0, 0);
74
75      frame_goban = newwin(GOBAN_SCREEN_HEIGHT + 2, GOBAN_SCREEN_WIDTH + 2, 0, 0);
76      win_goban = newwin(GOBAN_SCREEN_HEIGHT, GOBAN_SCREEN_WIDTH, 1, 1);
77      box(frame_goban, '|', '-');
78      scrollok(win_goban, FALSE);
79      wmove(win_goban, 0, 0);
80
81      cbreak();
82      noecho();
83
84      memcpy(goban_plane_orig, goban_plane, sizeof(goban_plane));
85      init_goban();
86
87      wrefresh(frame_info);
88      wrefresh(win_info);
89      wrefresh(frame_goban);
90      wrefresh(win_goban);
91  }
92
93  void session_loop()
94  {
95      int c;
96      fd_set readOk;
97      int i;
98      int y, x;
99      char message[BUF_LEN];
100     int status;
101     int is_game_loop   = 1;
102     int is_game_finish = 0;
103     int game_step = 0;
104
105     while (1) {
106         readOk = mask;
```

```
107         select(width, (fd_set *)&readOk, NULL, NULL, NULL);
108
109         if (FD_ISSET(0, &readOk)) {
110             c = getchar();
111             getyx(win_goban, y, x);
112             switch (c) {
113             case 'j':
114                 wmove(win_goban, y+1, x);
115                 break;
116             case 'k':
117                 wmove(win_goban, y-1, x);
118                 break;
119             case 'h':
120                 wmove(win_goban, y, x-2);
121                 break;
122             case 'l':
123                 wmove(win_goban, y, x+2);
124                 break;
125             case ' ':
126                 if (is_game_finish) break;
127                 if (!is_my_turn(game_step, goban_my_stone)) break;
128                 if (!put_stone(y, x, goban_my_stone)) break;
129
130                 sprintf(send_buf, "(%d,%d) %c\n", x, y, goban_my_stone);
131                 write(session_soc, send_buf, strlen(send_buf));
132
133                 break;
134             case 'r':
135             case 'c':
136                 sprintf(send_buf, "reset\n");
137                 write(session_soc, send_buf, strlen(send_buf));
138                 break;
139             case 'q':
140                 sprintf(send_buf, "quit\n");
141                 write(session_soc, send_buf, strlen(send_buf));
142                 break;
143             }
144             wrefresh(win_info);
145             wrefresh(win_goban);
146         }
147
148         if (FD_ISSET(session_soc, &readOk)) {
149             status = read(session_soc, recv_buf, BUF_LEN);
150             if (recv_buf[0] == ':') {
151                 // Game start!
152                 int id;
153                 sscanf(recv_buf, ":%d", &id);
154                 if (id == 0) {
155                     goban_my_stone = 'x';
156                     goban_peer_stone = 'o';
157                     strcpy(message, "Wait.");
158                 } else {
159                     goban_my_stone = 'o';
160                     goban_peer_stone = 'x';
161                     strcpy(message, "It's your turn!");
162                 }
163                 sprintf(recv_buf, "Game start! %s\n", message);
164                 werase(win_info);
```

```
165                 waddstr(win_info, recv_buf);
166             }
167             else if (recv_buf[0] == '(') {
168                 // Player put stone.
169                 char stone_char;
170                 sscanf(recv_buf, "(%d,%d) %c", &x, &y, &stone_char);
171                 put_stone(y, x, stone_char);
172                 game_step++;
173                 if ((status = is_my_turn(game_step, goban_my_stone)) > 0) {
174                     sprintf(message, "It's your turn! (remains: %d)\n", status);
175                 } else {
176                     sprintf(message, "%s\n", "Wait");
177                 }
178                 werase(win_info);
179                 waddstr(win_info, message);
180 
181                 if (stone_char == goban_my_stone && detect_rokumoku(stone_char)) {
182                     werase(win_info);
183                     waddstr(win_info, "You win!");
184                     is_game_finish = 1;
185                 }
186                 if (stone_char == goban_peer_stone && detect_rokumoku(stone_char)) {
187                     werase(win_info);
188                     waddstr(win_info, "You lose!");
189                     is_game_finish = 1;
190                 }
191             }
192             else if (strstr(recv_buf, "reset") != NULL) {
193                 // Reset game.
194                 init_goban();
195                 game_step = 0;
196                 is_game_finish = 0;
197                 if (goban_my_stone == 'x') {
198                     strcpy(message, "Wait.");
199                 } else {
200                     strcpy(message, "It's your turn!");
201                 }
202                 sprintf(recv_buf, "Game start! %s\n", message);
203                 werase(win_info);
204                 waddstr(win_info, recv_buf);
205             }
206             else if (strstr(recv_buf, "quit") != NULL) {
207                 // Quit game.
208                 is_game_loop = 0;
209             }
210             else {
211                 // Received broadcast message.
212                 werase(win_info);
213                 waddstr(win_info, recv_buf);
214             }
215 
216             wrefresh(win_info);
217             wrefresh(win_goban);
218         }
219 
220         if (is_game_loop == 0) break;
221     }
222 
```

```
223     die();
224 }
225 
226 static void init_goban()
227 {
228     int x, y;
229     memcpy(goban_plane, goban_plane_orig, sizeof(goban_plane_orig));
230 
231     wclear(win_goban);
232     x = 0;
233     for (y = 0; y < GOBAN_SCREEN_HEIGHT; y++) {
234         wmove(win_goban, y, x);
235         waddstr(win_goban, goban_plane[y]);
236     }
237     wmove(win_goban, GOBAN_SCREEN_HEIGHT/2, GOBAN_SCREEN_WIDTH/2);
238 }
239 
240 // Return true if it's my turn.
241 // game_step: 0 1 2 3 4 5 6 7 8 9 10 ...
242 // stone:     o x x o o x x o o x x  ...
243 static int is_my_turn(int game_step, char stone_char)
244 {
245     int mod;
246     if (stone_char == 'o' && game_step == 0) return 1;
247     if (stone_char == 'x' && game_step == 0) return 0;
248     mod = (game_step - 1) % 4;
249     if (stone_char == 'o' && mod == 2) return 2;
250     if (stone_char == 'o' && mod == 3) return 1;
251     if (stone_char == 'x' && mod == 0) return 2;
252     if (stone_char == 'x' && mod == 1) return 1;
253     return 0;
254 }
255 
256 static int put_stone(int y, int x, char stone_char)
257 {
258     if (goban_plane[y][x] != '.') return 0;
259     goban_plane[y][x] = stone_char;
260 
261     wmove(win_goban, y, x);
262     waddch(win_goban, stone_char);
263     wmove(win_goban, y, x);
264     return 1;
265 }
266 
267 static void die()
268 {
269     endwin();
270     close(session_soc);
271     exit(0);
272 }
273 
274 static int detect_rokumoku(char stone_char)
275 {
276     int cnt = 0;
277     int cnt2 = 0;
278     int x, y;
279     int k;
280     for (y = 0; y < GOBAN_SCREEN_HEIGHT; y++) {
```

```
281         cnt = 0;
282         for (x = 0; x < GOBAN_SCREEN_WIDTH - 1; x += 2) {
283             cnt = (goban_plane[y][x] == stone_char) ? (cnt + 1) : 0;
284             if (cnt == 6) return 1;
285         }
286     }
287
288     for (x = 0; x < GOBAN_SCREEN_WIDTH - 1; x += 2) {
289         cnt = 0;
290         for (y = 0; y < GOBAN_SCREEN_HEIGHT; y++) {
291             cnt = (goban_plane[y][x] == stone_char) ? (cnt + 1) : 0;
292             if (cnt == 6) return 1;
293         }
294     }
295
296     for (y = 0; y < GOBAN_SCREEN_HEIGHT; y++) {
297         for (x = 0; x < GOBAN_SCREEN_WIDTH - 1; x += 2) {
298             cnt = 0;
299             cnt2 = 0;
300             for (k = 0; k < GOBAN_SCREEN_WIDTH / 2 - 1; k++) {
301                 if (!(y + k >= 0 && y + k < GOBAN_SCREEN_HEIGHT)) continue;
302                 if (!(y + k * 2 < GOBAN_SCREEN_WIDTH - 1)) continue;
303                 if (!(GOBAN_SCREEN_WIDTH - 2 - x - k * 2 >= 0)) continue;
304                 cnt  = (goban_plane[y + k][x + k * 2] == stone_char) ? (cnt + 1) :
                        0;
305                 cnt2 = (goban_plane[y + k][GOBAN_SCREEN_WIDTH - 2 - x - k * 2] ==
                        stone_char) ? (cnt2 + 1) : 0;
306                 if (cnt == 6) return 1;
307                 if (cnt2 == 6) return 1;
308             }
309         }
310     }
311
312     return 0;
313 }
```

表 1　プレイヤーからのキーの入力に対応する処理

キーの入力	行う処理
j	一つ下へカーソルを移動する
k	一つ上へカーソルを移動する
h	一つ左へカーソルを移動する
l	一つ右へカーソルを移動する
space	現在のカーソルの位置に石を置く
r	ゲームをリセットする
q	ゲームを終了する

表 2　サーバから送られてきた文字列とそれに対応する処理

送られてきた文字列	行う処理
:N	数字 N が 1 なら先攻、0 なら後攻でゲーム開始
(X,Y) C	座標 (X,Y) の位置に文字 C の石を置く
reset	ゲームをリセットする
quit	ゲームを終了する

server.c であるリスト 5 では、サーバ側のプログラムを立ち上げた時にクライアント側からの通信を待ち、2 台のクライアントが接続したらゲームを開始するなどの一連の作業を行う。

リスト 5　server.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "sessionman.h"
#include "mylib.h"

int main(int argc, char const *argv[]) {
    int num;
    int soc;
    int maxfd;

    num = 2; // player count

    if ((soc = mserver_socket(PORT, num)) == -1) {
        fprintf(stderr, "cannot setup server\n");
        exit(1);
    }

    maxfd = mserver_maccept(soc, num, enter);

    sessionman_init(num, maxfd);

    sessionman_loop();

    return 0;
}
```

client.c であるリスト 6 では、クライアント側のプログラムを立ち上げた時にサーバとソケット通信をするための処理を行い、ゲームでのレンダリングやユーザ入力の処理などの一連の作業を行う。

リスト 6　client.c

```c
#include "session.h"

int main(int argc, char const *argv[]) {
    int soc;
    char hostname[HOSTNAME_LENGTH];

    printf("Input sever's hostname: ");
    fgets(hostname, HOSTNAME_LENGTH, stdin);
    chop_newline(hostname, HOSTNAME_LENGTH);

```

```
11      if ((soc = setup_client(hostname, PORT)) == -1) {
12          exit(1);
13      }
14
15      session_init(soc);
16
17      session_loop();
18
19      return 0;
20  }
```

最後に、作成したプログラムをコンパイルする Makefile をリスト 7 に示す。リスト 7 の Makefile を使って make すると、bin ディレクトリの下に s と c という実行ファイルができる。なお、s と c はそれぞれサーバ用とクライアント用の実行ファイルであることを表している。

リスト 7　Makefile

```
1  MYLIBDIR = mylib
2  MYLIB    = $(MYLIBDIR)/mylib.a
3  OBJS1    = server.o sessionman.o
4  OBJS2    = client.o session.o
5  CFLAGS   = -I$(MYLIBDIR)
6
7  all: bin bin/s bin/c
8
9  bin:
10   mkdir $@
11
12 bin/s: $(OBJS1)
13   $(CC) -o $@ $^ $(MYLIB) -lncurses
14
15 bin/c: $(OBJS2)
16   $(CC) -o $@ $^ $(MYLIB) -lncurses
17
18 server.o: sessionman.h
19 client.o: session.h
20
21 clean:
22   $(RM) bin/s bin/c $(OBJS1) $(OBJS2) *~
```

3　実行結果

まずサーバ側で bin/s を実行してからクライアント側となる 2 つ画面を用意して, それぞれで bin/c を実行した結果を図 1 と 2 に示す。

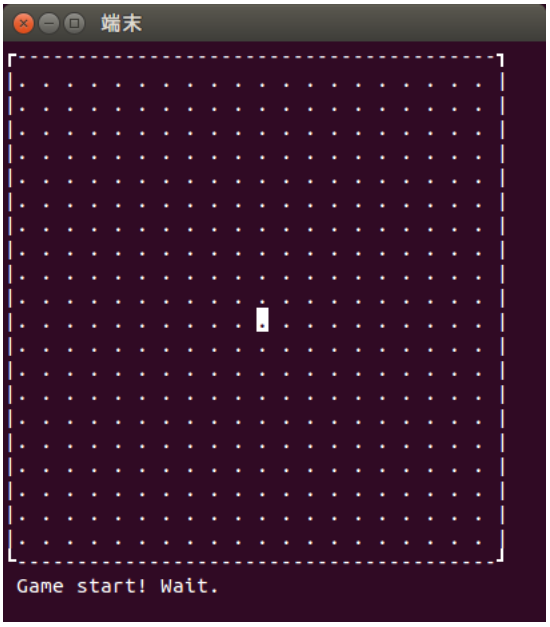

図 1　Caption

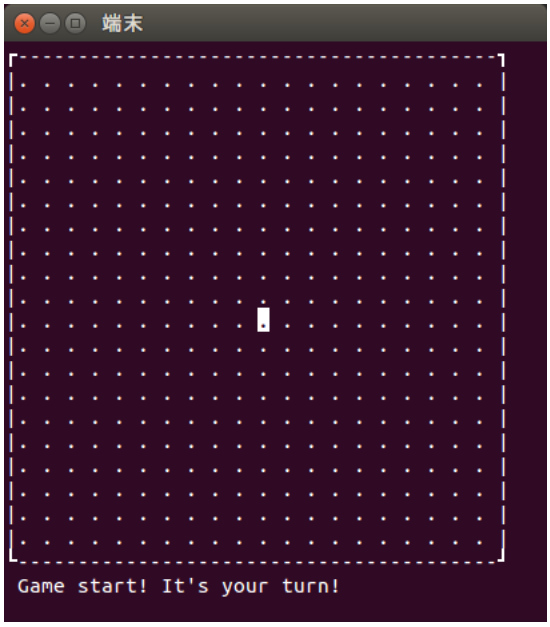

図 2　Caption

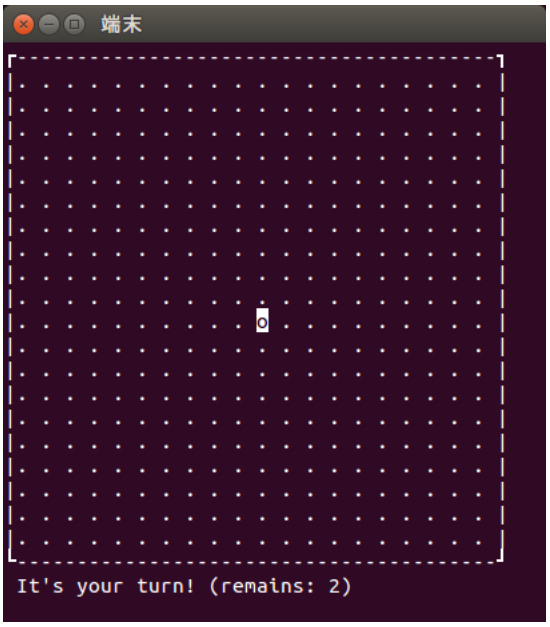

図 3　Caption

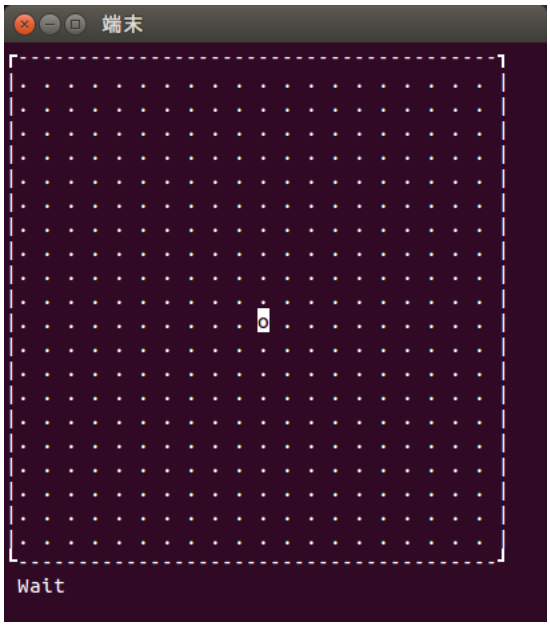

図 4　Caption

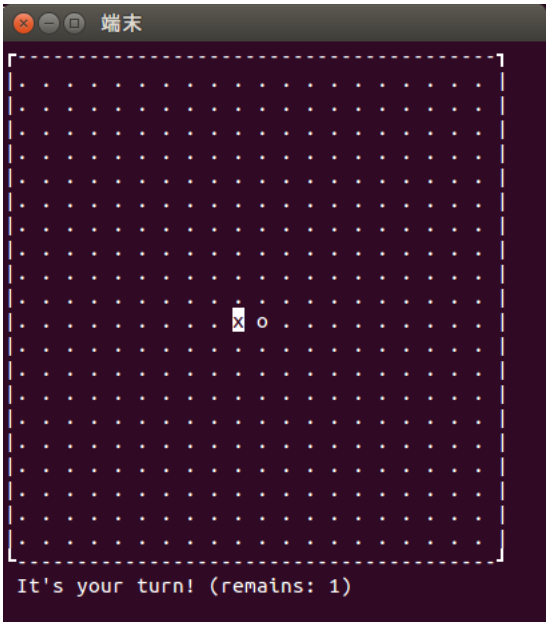

図 5　Caption

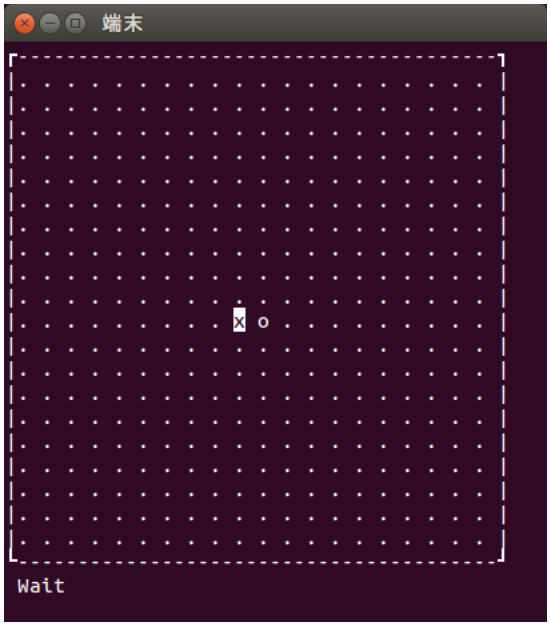

図 6　Caption

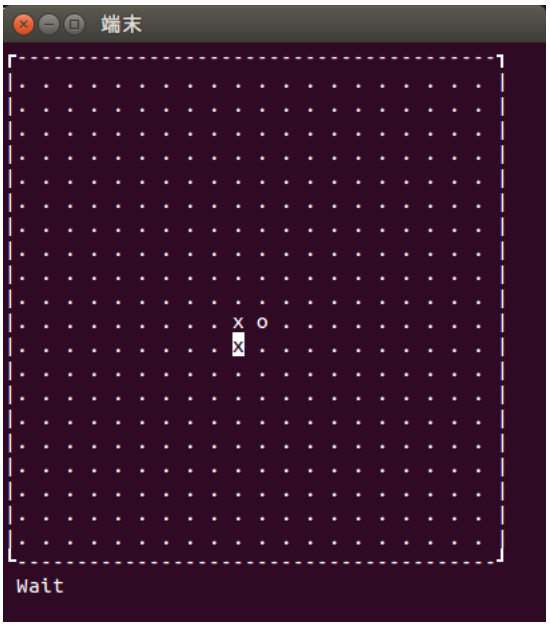

図 7　Caption

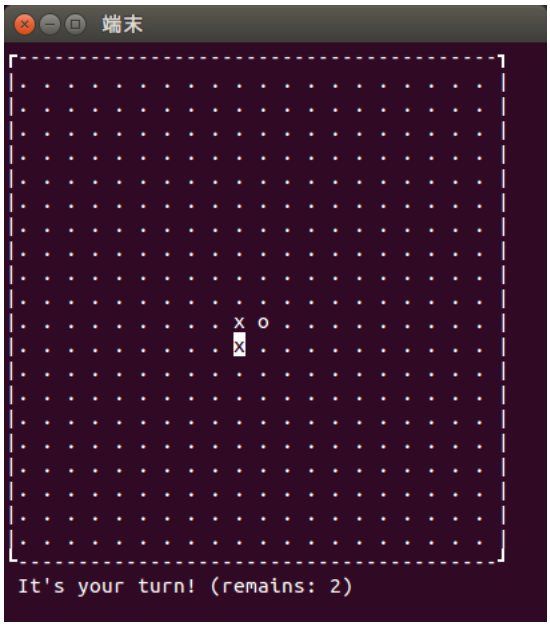

図 8　Caption

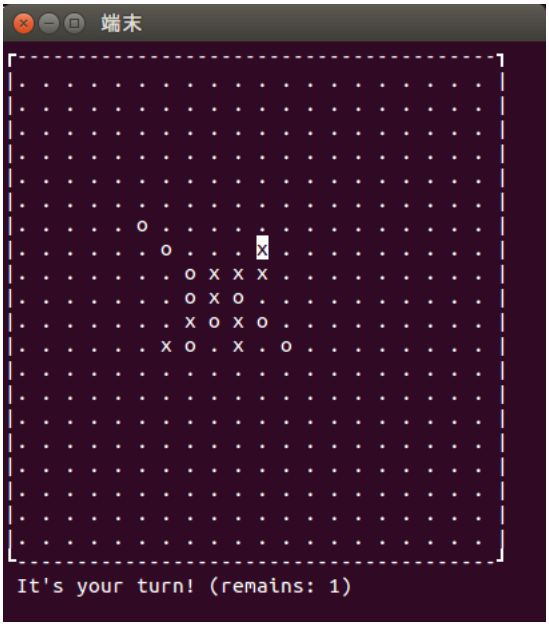

図 9　Caption

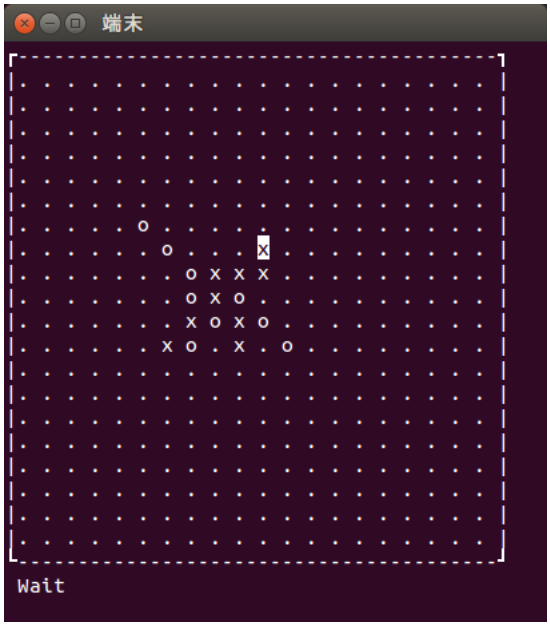

図 10　Caption

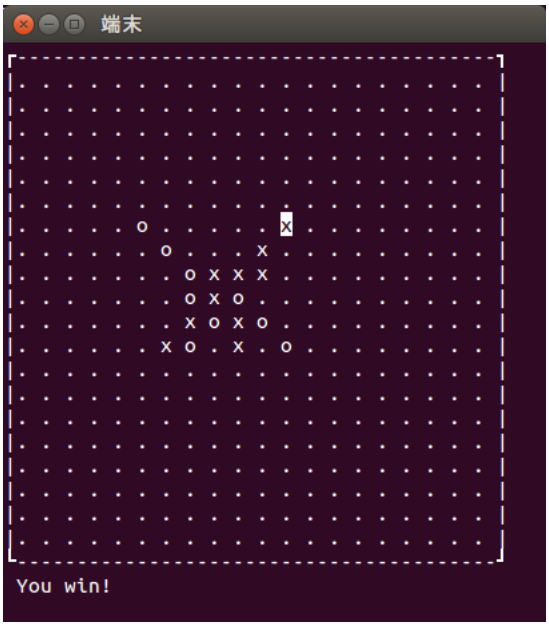

図 11　Caption

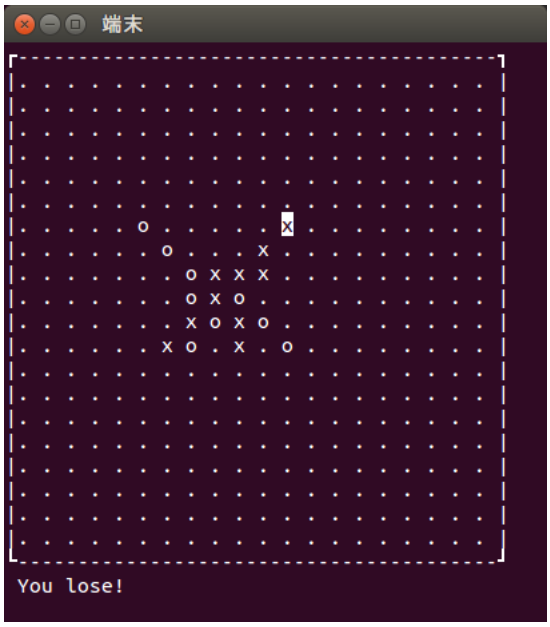

図 12　Caption

4　考察