Практикум на ЭВМ
client-server

Баев А.Ж.

Казахстанский филиал МГУ

19 ноября 2019

Семестровая работа.

1. Сервер
- ждет подключения N игроков
- рассылает всем игрокам стартовую карту
- обрабатывает ходы игроков

2. Клиент
- подключается к серверу
- ждет стартовую карту
- выполняет ходы

include

```
1 #include <unistd.h>
2 #include <netdb.h>
3 #include <sys/types.h>
4 #include <sys/socket.h>
5 #include <netinet/in.h>
6 #include <arpa/inet.h>
```

Сервер. Создание сокета

```
1  //open socket, return socket descriptor
2  int server_socket = socket(PF_INET, SOCK_STREAM, 0);
3
4  //set socket option
5  int socket_option = 1;
6  setsockopt(server_socket,
7             SOL_SOCKET,
8             SO_REUSEADDR,
9             &socket_option,
10            sizeof(socket_option));
```

Сервер. Привязка адреса

```
1  //set socket address
2  struct sockaddr_in server_address;
3  server_address.sin_family = AF_INET;
4  server_address.sin_port = htons(8080);
5  server_address.sin_addr.s_addr = INADDR_ANY;
6  bind(server_socket,
7       (struct sockaddr *) &server_address,
8       sizeof(server_address));
9
10 //start listen mode
11 listen(server_socket, 5);
```

Сервер. Подключения

```
1  while(1) {
2      struct sockaddr_in client;
3      socklen_t size;
4      int client_socket;
5      client_socket = accept(server_socket,
6                             (struct sockaddr *) &client
7                             &size);
8      char *addr = inet_ntoa(client.sin_addr);
9      int port = ntohs(client.sin_port);
10
11     printf("connected: %s %d\n", addr, port);
12
13     //write(client_socket, ...)
14
15     close(client_socket);
16 }
```

Клиент. Создание сокета

```
1  //open socket, return socket descriptor
2  int server_socket = socket(PF_INET, SOCK_STREAM, 0);
3
4  //prepare server address
5  struct hostent *host = gethostbyname(ip);
6  struct sockaddr_in server_address;
7  server_address.sin_family = AF_INET;
8  server_address.sin_port = htons(port);
9  memcpy(&server_address.sin_addr,
10         host -> h_addr_list[0],
11         sizeof(server_address));
```

Клиент. Подключение

```
1  //connect
2  connect(server_socket,
3          (struct sockaddr *) &server_address,
4          sizeof(server_address));
5
6  //read(server_socket, ...)
```

Неканонический терминал (библиотека ncurses)

gcc prog.c -lcurses

```
1  #include <curses.h>
2  #include <locale.h>
3  int main(void) {
4      int prev_x = 0, prev_y = 0, caret_y = 0, caret_x =
5      int flag = 1;
6      setlocale(LC_ALL, "");
7      if (!initscr()) return 1;
8      cbreak();
9      noecho();
10     nonl();
11     meta(stdscr, TRUE);
12     intrflush(stdscr, FALSE);
13     keypad(stdscr, TRUE);
14
15     if (has_colors()) {
16         start_color();
17         init_pair(1, COLOR_WHITE, COLOR_BLUE);
18     }
```

Неканонический терминал (библиотека ncurses)

```
1     attrset(COLOR_PAIR(1));
2     bkgdset(COLOR_PAIR(1));
3     clear();
4     while (flag) {
5        mvaddch(prev_y, prev_x, ' ');
6        mvaddch(caret_y, caret_x, '*');
7        move(caret_y, caret_x);
8        refresh();
9        prev_x = caret_x;
10       prev_y = caret_y;
```

Неканонический терминал (библиотека ncurses)

```
1        c = getch();
2        switch (c) {
3        case 033:
4            flag = 0;
5            break;
6        case KEY_UP:
7            if (caret_y > 0) caret_y--;
8            break;
9        case KEY_DOWN:
10           if (caret_y < LINES - 1) caret_y++;
11           break;
12       case KEY_LEFT:
13           if (caret_x > 0) caret_x--;
14           break;
15       case KEY_RIGHT:
16           if (caret_x < COLS - 2) caret_x++;
17           break;
18       }
19   }
```

Неканонический терминал (библиотека ncurses)

```
1       bkgdset(COLOR_PAIR(0));
2       clear();
3       refresh();
4       endwin();
5       return 0;
6   }
```

Задание

Задание практикума: сетевая игра, консольный Doom

Алексей Сальников
1) Требуется реализовать игровой сервер, игровой клиент, отображатель статистики.
2) Сервер и клиенты должны взаимодействовать через сеть путём установки TCP соединения.
3) При своём запуске сервер читает файл с картой (имя файла указывается в параметрах при запуске).
4) Клиент соединятся с сервером (hostname сервера и номер TCP порта сервера указываются клиенту в аргументах main при запуске

Задание

Игра происходит в прямоугольном лабиринте, представленным как набор точек некоторой матрицы размера $M \times N$. Лабиринт вместе с его размером должны быть заданы в файле карты, при этом сам файл карты должен иметь текстовое представление легко читаемое человеком. Цель игры оказаться выжившим в лабиринте с максимальным уровнем здоровья.

```
10 20
##########################
# #  #   #             #
# #   #  #  #####      #
# ##### #      #      #
#       ######### ###
#  # ##      #       #
#  ########  # ##### #
## ##        # #     #
## ### ############## #
#  #        #     #  #
#       #      #     #
##########################
```

Задание

Лабиринт состоит из стенок, аптечек/потравлялок, и проходов. По точке, содержащей аптечку/потравлялку можно двигаться, по стенкам и за границами лабиринта двигаться нельзя. Аптечка обладает лечебным/отравляющим эффектом определённой силы. Игрок может съесть аптечку/потравлялку, при этом к его здоровью прибавляется/отнимается численное значение эффекта. По внешнему виду аптечки/потравлялки нельзя ничего сказать о силе её воздействия и о знаке её воздействия. После употребления аптечки соответствующая клетка считается просто проходом (повторно съесть аптечку нельзя). В лабиринте могут встречаться другие игроки, которые всегда занимают одну точку пространства (в стенке игрок не может находиться).
В начальный момент игроки расставляются сервером на карту случайным образом. Они должны обязательно оказаться в допустимой точке (то есть не на стенке и не на одной клетке с другим игроком).