VENOM

Venom is a lightweight and simple binary protocol over TCP or UDP, created for game snake as a credit project for the subject of programming network applications UMCS Summer 2021.

Documentation

Venom was designed to be easy to use and friendly for developers. If you want to send or receive messages you need to import four classes from the library: **MessageType**, **Header**, **Body** and **Message**.

MessageType is enum contains values to determine type of messages:

```
LOGIN_CLIENT = 0x01
LOGIN_SERVER = 0x02
LIST_GAMES_CLIENT = 0x03
LIST_GAMES_SERVER = 0x04
CREATE_GAME_CLIENT = 0x05
CREATE_GAME_SERVER = 0x06
JOIN_GAME_CLIENT = 0x07
JOIN_GAME_SERVER = 0x08
EXIT_GAME_CLIENT = 0x09
EXIT_GAME_SERVER = 0x0a
SEND_MOVE = 0x0b
SEND_STATE = 0x0c
UNKNOWN = 0x0d
```

Next you need to create two objects from classes Header and Body, for example:

```
header = Header(sender=0, message_type=MessageType.SEND_STATE)
body = Body()
```

```
sender=0
```

This means that it is a message from the server, if you are sending a message from client to server, set sender to 1.

Object of the Body class has dictionary **data,** where you must provide relevant variables, different for each type of message:

MessageType.SEND_STATE

```
            body.data["operation_success"] = b'\x20'
            body.data["game_id"] = 10000
            body.data["p1_direction"] = "d".encode("ascii")
            body.data["p2_direction"] = "d".encode("ascii")
            body.data["p1_snake"] = [(1, 2), (1, 3)]
            body.data["p2_snake"] = [(1, 2), (1, 3)]
            body.data["food"] = (1, 5)
            body.data["pt1"] = 5
            body.data["pt2"] = 3
            body.data["players_num"] = 2
            body.data["p1_over"] = False
            body.data["p2_over"] = False
```

Where:

- game_id is integer
- p1/p2_direction are encoded one byte strings
- p1/p2 snake are lists of tuples that contain two elements x and y
- food is tuple of two values x and y
- pt1/pt2 are integers
- players_num is integer
- p1/p2_over are booleans

MessageType.SEND_MOVE

```
body.data["user_id"] = 10001
body.data["game_id"] = 10000
body.data["move"] = "u".encode("ascii")
```

Where:

- user_id is integer
- game_id is integer
- move is encoded one byte string

MessageType.LOGIN_CLIENT

```
body.data["nickname"] = "snake"
```

Where:

- nickname is string with length < 255

MessageType.LOGIN_SERVER

```
body.data["operation_success"] = True
body.data["user_id"] = 5
```

Where:

- operation_success is boolean
- user_id is integer

MessageType.LIST_GAMES_CLIENT

```
body.data["user_id"] = 5
```

Where:

- user_id is integer

MessageType.LIST_GAMES_SERVER

```
for game_id in self.games:
            can_join = 1 if self.games[game_id]["players_num"] < 2 else 0
            game_name = self.games[game_id]["game_name"]
            game_info_list.append({"game_id": game_id, "can_join":
can_join, "game_name": game_name})

        body = Body()
        header = Header(sender=0,
message_type=MessageType.LIST_GAMES_SERVER)
        body.data["operation_success"] = True
        body.data["games"] = game_info_list
```

MessageType.JOIN_GAME_CLIENT

```
body.data["user_id"] = 1
body.data["game_id"] = 10000
```

MessageType.JOIN_GAME_SERVER

```
body.data["operation_success"] = 0x20
body.data["is_player_1"] = True
```

MessageType.CREATE_GAME_CLIENT

```
body.data["user_id"] = 10000
body.data["game_name"] = "first_game"
```

MessageType.CREATE_GAME_SERVER

```
body.data["game_id"] = 10005
```

MessageType.EXIT_GAME_CLIENT

```
body.data["user_id"] = 1
body.data["game_id"] = 10000
```

MessageType.EXIT_GAME_SERVER

```
body.data["operation_success"] = True
```

Create object of type **Message,** with two objects created before**:**

```
message = Message(header=header, body=body)
```

And use **to_bytes()** method to get encoded message ready to send:

```
message.to_bytes()
```

If you want to receive message, just use static method **from_bytes()** in **Message** class:

```
msg = Message.from_bytes(data)
```

Where data is bytes object received from the sender.

Created object will have sended informations in **body.data.**

Specification

Header

Header includes two bytes:

- First byte contains information about the sender, there are two values possible 0 if server, 1 if client.
- Second byte contains information about the message type, next bytes are defined by this type.

Message types

- **Register / Login.**

  Client message (value 0x01): After the header there is one byte which specifies nickname length x, next there are x bytes: nickname in ASCII.

  Server response message (value 0x02): Server responses with header, byte informing if operation was successful: value 0x20 if success 0x50 otherwise and 2 bytes with unique user ID if successful.
- **List games**.

  Client message (value 0x03): Header and two bytes with unique user ID.

  Server response message (value 0x04): Header, two bytes with number of games x, then x games contains: 2 bytes game ID, byte determines if you can join this game, byte with game name length y, and y bytes with game name.
- **Create game.**

Client message (value 0x05): two bytes with user_id and game_name_len and next game_name_len bytes of game_name.

Server response message (value 0x06): return game id, two bytes

- **Join game.**

  Request (value 0x07): Two bytes: game id and two bytes: user id

  Response (value 0x08): response with information that operation was successful 0x20 if success, 0x50 otherwise, and information if player is player_1

- **Exit game.**

  Request (value 0x09): game id (two bytes), user id (two bytes)

  Response (value 0x0a): operation success

- **Send move.**

  Client message (value 0x0b): Two bytes unique user ID, two bytes unique game ID, one byte move in ASCII.

- **Send game state.**

  Server message (value 0x0c): Header, game_id (two bytes), p1_direction (one byte), p2_direction (one byte), food x and y (four bytes), points_1 (two bytes), points_2 (two bytes), players_num (two bytes), p1 and p2 over (two bytes), p1_snake_len (two bytes), p2_snake_len (two bytes), 4 * p1_snake_len ([(x, y), (x, y)...]) for two snakes.