```
#ifndef TERMINALPROTOCOL_H
#define TERMINALPROTOCOL_H

#include "Utils/Time.h"

#define STR_LENGTH 30

// Requête transmise/reçue via le réseau.
typedef struct _terminal_protocol {
    // Type de requête
    enum protocol_type {
        ACK, FAIL, LOGIN, NEXT_DEPARTURE, DEPARTURE_KNOWN, DEPARTURE_UNKNOWN,
        NO_FERRY, BEGIN_LOADING, END_LOADING, FERRY_LEAVING, ASK_FOR_FERRY,
        FERRY_RESERVED, FERRY_ARRIVING, CLOSE
    } type;

    // Contenu de la requête
    union protocol_content {
        // Demande la connexion avec un utilisateur et un mot de passe
        struct login_protocol {
            char user[STR_LENGTH];
            char password[STR_LENGTH];
            int terminal_id;
        } login;

        // Donne l'heure du prochain départ
        s_time departure_known;

        // Demande l'autorisation du début de l'embarquement
        s_time begin_loading;

        // Notifie la fin de l'embarquement et demande l'autorisation de partir
        s_time end_loading;

        // Notifie que le ferry quitte le terminal
        s_time ferry_leaving;

        // Donne le nom du ferry qui est attribué au terminal
        int ferry_reserved;

        // Notifie l'arrivée d'un ferry au terminal
        struct ferry_arriving_protocol {
            s_time time;
            int ferry_id;
        } ferry_arriving;

        // Notifie l'heure de fermeture
        s_time close;
    } content;
} terminal_protocol;

// Envoie un packet qui ne contient aucune information si ce n'est son type.
inline void send_flag_packet(ClientSocket sock, const terminal_protocol::protocol_type type)
{
    terminal_protocol packet;
    packet.type = type;
    sock.send<terminal_protocol>(&packet);
}

#endif // TERMINALPROTOCOL_H
```