```
#ifndef SOCKETCLIENT_H
#define SOCKETCLIENT_H

#include <arpa/inet.h>
#include <sys/socket.h>
#include <unistd.h>

#include <iostream>
#include <vector>

#include "SocketsUtils.h"
#include "SocketException.h"

extern pthread_mutex_t mutex_pause;

using namespace std;

class ClientSocket
{
private:
    int _socket;

public:
    ClientSocket(const char* ip, const int port);
    ClientSocket(const int socket_fd);
    ClientSocket(const ClientSocket& other);
    ~ClientSocket() {};
    ClientSocket& operator=(const ClientSocket& other);

    int get_socket_fd() const { return this->_socket; }

    // Envoie une donnée
    template <class T>
    inline ssize_t send(const T *data)
    {
        return this->send<T>(data, 1);
    }

    // Envoie un vecteur de données
    template <class T>
    inline ssize_t send(const void *data, size_t length)
    {
        ssize_t total = 0;
        while (total < (ssize_t) (sizeof (T) * length)) {
            ssize_t ret = socket_utils::send(
                this->_socket, (const char *) data + total,
                length * sizeof (T) - total, 0
            );

            if (ret == -1)
                throw SocketException("Envoi incomplet des donnees");

            total += ret;
        }

        return total;
    }

    inline void send_string(char *data)
    {

        char c;
        do {
            c = *data;
            this->send<char>(&c);
            data++;
        } while (c != '\0');
    }
    // Reçoit une donnée
    template <class T>
    inline ssize_t receive(T *data)
    {
        return this->receive<T>(data, 1);
    }

    // Reçoit un vecteur de données
    template <class T>
    inline ssize_t receive(void *data, size_t length)
    {
        ssize_t total = 0;
        while (total < (ssize_t) (sizeof (T) * length)) {
            // Vérifie que le serveur n'est pas en pause
            pthread_mutex_lock(&mutex_pause);
            pthread_mutex_unlock(&mutex_pause);

            ssize_t ret = socket_utils::recv(
                this->_socket, (char *) data + total,
                sizeof (T) * length - total, 0
            );

            if (ret == -1)
                throw SocketException("Reception incomplete des donnees");

            total += ret;
        }

        return total;
    }

    inline void receive_string(char *data)
    {
        char c;
        do {
            this->receive<char>(&c);
            *data = c;
            data++;
        } while (c != '\0');
    }

    void close();
};

#endif // SOCKETCLIENT_H
```