```cpp
#include "SocketsUtils.h"

#include <stdio.h>

namespace socket_utils
{
    // Initialise un socket. Utilisé par les classes ClientSocket et ServerSocket
    int init_socket(const unsigned int ip, const int port, socket_type type)
    {
        int _socket = socket_utils::socket(AF_INET, SOCK_STREAM, 0);

        struct sockaddr_in dest = {0};
        dest.sin_addr.s_addr = ip;
        dest.sin_port = htons(port);
        dest.sin_family = AF_INET;

        if (type == SERVER_SOCKET) {
            socket_utils::bind(
                _socket, (struct sockaddr*) &dest, sizeof (struct sockaddr_in)
            );
        } else {
            socket_utils::connect(
                _socket, (struct sockaddr*) &dest, sizeof (struct sockaddr_in)
            );
        }

        return _socket;
    }

    // Fonctions POSIX avec gestion des exceptions

    int socket(int domain, int type, int protocol)
    {
        int _socket;
        _socket = ::socket(domain, type, protocol);

        if (_socket == -1)
            throw SocketException("Erreur lors de l'initialisation de la socket");
        return _socket;
    }

    int connect(int _socket, const struct sockaddr *address,
                socklen_t address_len)
    {
        if (::connect(_socket, address, address_len) == -1)
            throw SocketException("Erreur lors de la connexion de la socket client");
        return 0;
    }

    int bind(int _socket, const struct sockaddr *address, socklen_t address_len)
    {
        if (::bind(_socket, address, address_len) == -1)
            throw SocketException("Erreur lors de la connexion de la socket serveur");
        return 0;
    }

    int listen(int _socket, int backlog)
    {
        if (::listen(_socket, backlog) == -1)
            throw SocketException("Erreur lors du lancement de l'écoute de la socket serveur");
        return 0;
    }

    int accept(int _socket, struct sockaddr *address, socklen_t *address_len)
    {
        int fd = ::accept(_socket, address, address_len);
        if (fd == -1)
            throw SocketException("Erreur lors de l'acceptation d'une connexion cliente");
        return fd;
    }

    ssize_t send(int _socket, const void *buffer, size_t length, int flags) {
        ssize_t ret = ::send(_socket, buffer, length, flags);
        if (ret == -1)
            throw SocketException("Erreur lors de l'envoi des données");
        return ret;
    }

    ssize_t recv(int _socket, void *buffer, size_t length, int flags)
    {
        ssize_t ret = ::recv(_socket, buffer, length, flags);
        if (ret == -1)
            throw SocketException("Erreur lors de la reception des données");
        return ret;
    }

    int shutdown(int _socket, int how)
```

```
    {
        if (::shutdown(_socket, how) == -1)
            throw SocketException("Erreur lors de la fermeture du socket");
        return 0;
    }

    int close(int fildes)
    {
        if (::close(fildes) == -1)
            throw SocketException("Erreur lors de la fermeture du descripteur de fichier");
        return 0;
    }
}
```