**Что необходимо сделать для запуска и работы системы FailuresTracker:**

Допустим, у вас есть несколько устройств, общающихся через разные интерфейсам передачи данных – например, IP-камера – TCP-сокет, файловый сервер – FTP-протокол, веб-страница – WebSocket. Вы хотите организовать передачу данных между любыми двумя устройствами. Для упрощения этой задачи и была создана система FailuresTracker. Строго говоря, «внешнее устройство» - не обязательно реальное оборудование – это может быть любой источник или приёмник данных.

Вам необходимо реализовать DLL библиотеки, работающие с интерфейсами, через которые устройства могут передавать данные, указать путь к этим библиотекам в конфигурационном файле (config.cfg) и запустить программу.

**Библиотеки DLL должны содержать следующие функции:**

uint32 AcceptConnection() – библиотека принимает входящее соединение от внешнего устройства или, возможно, сама подключается к некоторому внешнему устройству. Очевидно, библиотека сохраняет некоторое внутреннее значение, позволяющее в дальнейшем работать с этим устройством. Например, для TCP сокетов – это просто целочисленная переменная. Каждому такому значению библиотека должна сопоставить уникальный идентификатор, который она и вернёт из этой функции. В дальнейших описаниях функций, входной аргумент с именем id – и есть этот самый идентификатор.

uint32 IsReceiver( uint32 id ) – возвращает 0, если устройство с идентификатором id будет передавать данные. 1 – если принимать. В случае, когда одно и то же устройство может и принимать и передавать данные, для него потребуется создать два идентификатора, которые будут возвращены соответственно из двух вызовов функции AcceptConnection

MID (Match Identified) – это некоторая строка-идентификатор, используемая для соединения устройств – данные с какого устройства передать какому устройству. Вероятнее всего, этот идентификатор будет задан непосредственно в реализации DLL библиотек, когда станет известно, что с чем требуется соединить. При каждом новом подключении (возврате из функции AcceptConnection), система сохраняет MID подключившегося устройства, а также проверяет, подходит ли к этому устройству уже сохранённый чей-то MID. Если это условие выполнено, система соединяет их.

void GetMID( uint32 id, char[] midBuffer ) – записывает в midBuffer Match ID, соответствующий устройству с идентификатором id

uint32 MatchMID( uint32 id, char[] mid ) – возвращает 1, если строка mid является подходящим Match ID для устройства с идентификатором id. Возвращает 0, если не является.

Например, очередной вызов AcceptConnection вернул некоторый id. После этого система делает следующее: проверяет, вернёт ли MatchMID( id, storeMIDs[i] ) 1, для какого-либо MatchID, сохранённого ранее в массиве storedMIDs. Если это происходит, происходит проверка в обратную сторону – для наёденного подходящего устройства c идентификатором id\_1, вызывается функция MatchMID( id\_1, newMID ), где newMID – Match ID, полученный вызовом GetMID( id, newMID ). Если обе проверки были успешно выполнены, создаётся соединение между устройствами id и id\_1. В противном случае, если хотя бы одна проверка не прошла, newMID просто добавляется в список сохранённых MatchID.

uint32 GetData( uint32 id, byte[] dataBuffer ) – принимает данные data, передаваемые источником с идентификатором id (источник определяется вызовом IsReceiver(id))

void DispatchData( uint32 id, byte[] data ) – передаёт приёмнику id данные data (источник определяется вызовом IsReceiver(id)).

void CloseConnection( uint32 id ) – закрывает соединение к устройству id.

void StopAccepting() – прерывает работу функции AcceptConnection. Функцию AcceptConnection система вызывает в цикле, последовательно обрабатывая новые входящие соединения. Если очередного подключения не происходит, AcceptConnection зависает. Чтобы выйти из зависания (это понадобится при завершении работы системы) – и используется StopAccepting()

void Terminate() – останавливает все внутренние потоки DLL и очищает выделенные для её работы ресурсы.

**Структура файла config.cfg:**

Первая строка указывает, откуда брать пути к DLL-файлам, может иметь одно их трёх значений:

CONSOLE – пути будут введены вручную из консоли.

THIS FILE – пути будут указаны ниже в этом файле

ANOTHER FILE – пути указаны в другом файле. В этом случае на следующей строке должен быть указан быть к файлу с информацией о местонахождении DLL библиотек.

Вторая строка, или третья в случае, когда значение в первой строке – ANOTHER FILE, указывает, запускать систему сразу или вручную из консоли. Может иметь одно из двух значений:

START IMMEDIATELY – в этом случае система будет запущена сразу же после получения информации о местонахождении DLL-файлов.

START MANUALLY – в этом случае систему необходимо запустить вручную из консоли командой start.

После этого, в случае, когда значение в первой строке – THIS FILE, на каждой из следующих строк указаны пути к DLL библиотекам.

В файле могут быть пустые строки, при чтении они игнорируются. А также к каждой непустой строке применяется функция Trim(), убирающая из начала и конца строки пробельные символы.

**Для остановки системы:**

Для остановки работающей системы ввести в консоли команду stop. Для перезапуска - команду start. Для выхода из консоли при выключенной системе – exit.