Cameron's networking library(TCP)

Note: This library uses the library zlib, and I am not, nor do I claim to the writer of it.

Network:

```
CAMSNETLIB int InitializeNetworking();
```

Initializes networking.

Remarks:
- Must be called before you can do anything(lol)

Return value:
- Zero:
  - When function succeeds.
- Non Zero:
  - See WSAStartup.

```
CAMSNETLIB int CleanupNetworking();
```

Cleans up networking.

Return value:
- Zero:
  - When function succeeds.
- Non Zero:
  - See WSACleanup.

Remarks:
- Must be called for every InitializeNetworking call.

Client:

```
CAMSNETLIB TCPClientInterface* CreateClient(cfunc func, void(*const
disconFunc)(), int compression = 9, void* obj = nullptr);
```

Parameters:
- Func – Pointer to a function with the following signature:
  - void function(void* manager, void* client, BYTE* data, DWORD nBytes, void* obj)
- disconFunc – Pointer to a function with the following signature:
  - void function()

- [optional] compression – This sets level what compression the client will compress data to send at. Value of 1-9.
- [optional] Obj - Pointer to a class object, that is passed to the msghandler function; it is mostly used in oop.

Creates, and initializes a client object.

Remarks:
- Must be called before you can do anything(lol)
- A call to DestroyClient is required.

Return value:
- TCPClientInterface*

```
CAMSNETLIB void DestroyClient(TCPClientInterface*& client);
```

Destroys the specified client object.

```
virtual bool Connect(const LIB_TCHAR* dest, const LIB_TCHAR* port, float
timeOut = 5.0f) = 0;
```

Attempts to connect to the destination(IP address or hostname).
It waits/blocks until either:
1) A successful connection has been established.
2) Timeout period has returned

Return value:
- True:
  - When function succeeds.
- False:
  - If client is already connected, or function fails.

```
virtual void Shutdown() = 0;
```

Immediately shuts down connection to server, and performs cleanup.
It waits/blocks until function returns.

```
virtual void Disconnect() = 0;
```

Shuts down connection to server, and performs cleanup.
It does not wait/block.

```
virtual bool RecvServData() = 0;
```

Initializes socket, and receiving thread and starts receiving data from server.

Remarks:

- Must be called before you can send any type of data to server.

Return value:

- True:
  - When function succeeds.
- False:
  - If client is not connected.

```
virtual HANDLE SendServData(const char* data, DWORD nBytes) = 0;
```

Sends data to connected server.

Remarks:

- Handle must be closed after this is called either with WaitAndCloseHandle, or CloseHandle.

Return value:

- A handle to created send thread.

```
virtual void SendMsg(char type, char message) = 0;
virtual void SendMsg(const std::tstring& user, char type, char message) = 0;
```

Sends a message to connected server in the format of TYPE, MESSAGE.
These functions are wrappers to SendServData.

```
virtual void Ping() = 0;
```

Pings the server, should be called in message handler.

```
virtual void SetFunction(cfunc function) = 0;
```

Sets the clients function/message handler, it is called whenever a message is received.

```
virtual bool IsConnected() const = 0;
```

Return value:

- True:
  - If connected.
- False:
  - If not connected.

```
virtual Socket& GetHost() = 0;
```

Returns a reference to the connected socket.

Return value:

- Socket&

```
virtual void* GetObj() const = 0;
```

Returns a pointer to the object you specified in constructor.

Return value:

- Void*

Server:

```
CAMSNETLIB TCPServInterface* CreateServer(sfunc func, customFunc conFunc,
customFunc disFunc, USHORT maxCon = 20, int compression = 9, float
pingInterval = 30.0f, void* obj = nullptr);
```

Parameters:

- Func - Pointer to a function with the following signature:
  - void function(void* manager, void* client, BYTE* data, DWORD nBytes, void* obj)
- conFunc - Pointer to a function with the following signature:
  - void function(ClientData* data)
- disFunc - Pointer to a function with the following signature:
  - void function(ClientData* data)
- [optional] maxCon – The maximum amount of clients the server can support before it sends (TYPE_CHANGE, MSG_CHANGE_SERVERFULL) to the connecting client.
- [optional] compression –The level of compression the server will compress data to send at. Value of 1-9.
- [optional] pingInterval – The frequency the server sends ping messages to connected clients, to keep them from timing out.

- [optional] Obj - Pointer to a class object, that is passed to the msghandler function; it is mostly used in oop.

Creates, and initializes a server object.

Remarks:
- Must be called before you can do anything(lol)
- A call to DestroyServer is required.

Return value:
- TCPServInterface*

```
CAMSNETLIB void DestroyServer(TCPServInterface*& server);
```

Destroys the specified server object.

```
virtual bool AllowConnections(const LIB_TCHAR* port) = 0;
```

Binds host socket, and creates a thread that waits for connections to the server.

Remarks:
- Must be called before you can send any type of data to server.

Return value:
- True:
  - If function succeeds.
- False:
  - If function has already been called.
  - If function fails.

```
virtual HANDLE SendClientData(const char* data, DWORD nBytes, Socket addr, bool single) = 0;
virtual HANDLE SendClientData(const char* data, DWORD nBytes, Socket* pcs, USHORT nPcs) = 0;
virtual HANDLE SendClientData(const char* data, DWORD nBytes, std::vector<Socket>& pcs) = 0;
```

Sends data to specified clients.

Remarks:

- Handle must be closed after this is called either with WaitAndCloseHandle, or CloseHandle.
- First overload, the value of single determines what the function does
  - If single is true it sends only to address specified.
  - If single is false, and addr is not connected, it sends to all clients currently connected to the server.
  - If single is false, and addr is connected, it sends to all clients, excluding the addr specified.

Return value:

- A handle to created send thread.

```
virtual void SendMsg(Socket pc, bool single, char type, char message) = 0;
virtual void SendMsg(Socket* pcs, USHORT nPcs, char type, char message) = 0;
virtual void SendMsg(std::vector<Socket>& pcs, char type, char message) = 0;
virtual void SendMsg(const std::tstring& user, char type, char message) = 0;
```

Sends a message to specified clients in the format of TYPE, MESSAGE.
These functions are wrappers to SendClientData.

```
virtual ClientData* FindClient(const std::tstring& user) const = 0;
```

Return value:

- A pointer to the ClientData, specified by user.

```
virtual void Shutdown() = 0;
```

Immediately shuts down all connections to server, and performs cleanup.
It waits/blocks until function returns.

```
virtual ClientData** GetClients() const = 0;
```

Return value:

- A pointer to the array of clients.

```
virtual USHORT ClientCount() const = 0;
```

Return value:

- Returns the number of connected clients.

```
virtual void SetPingInterval(float interval) = 0;
```

Sets the interval at which the server pings the clients.

```
virtual bool MaxClients() const = 0;
```

Return value:
- True:
  - If number of connected clients is at the maximum number of clients.
- False:
  - If number of connected clients is less than maximum clients.

```
virtual bool IsConnected() const = 0;
```

Return value:
- True:
  - If listening socket has been binded.
- False:
  - If listening socket has not been binded.

```
virtual Socket& GetHost() = 0;
```

Returns a reference to the connected socket.

Return value:
- Socket&

```
virtual void* GetObj() const = 0;
```

Returns a pointer to the object you specified in constructor.

Return value:
- Void*

Other:

```
CAMSNETLIB void WaitAndCloseHandle(HANDLE& hnd);
```

Waits for the specified handle to be triggered, then closes the specified handle.

Server and Client auto handled messages

Key:

-Checkmarks mean auto handled.

TYPE	MESSEAGE	SERVER	CLIENT	ADDITIONAL DATA
TYPE_PING (0)	MSG_PING (0)	Sent to client every X seconds. ✓	Needs to be handled in message handler.	NONE
TYPE_CHANGE (-128)	MSG_CHANGE_SERVERFULL (-128)	Sent to client when server is full. ✓	Needs to disconnect client in message handler.	NONE
TYPE_CHANGE (-128)	MSG_CHANGE_DISCONNECT (-127)	Sent to all clients when any user/pc disconnects ✓	Optionally handled on client.	const LIB_TCHAR* includes NULL char

Notice:

THIS IS A UNICODE BUILD ATTEMPTS TO USE MULTIBYTE/ASCII WILL RESULT IN A CRASH.