Windows Socket (WinSock) Programming Cheat Sheet

Required Headers and Library

```
#include <winsock2.h>
#include <ws2tcpip.h>
#pragma comment(lib, "ws2_32.lib")  // Link with ws2_32.lib
```

WinSock Initialization

```
WSADATA wsaData;
int result = WSAStartup(MAKEWORD(2, 2), &wsaData);
if (result != 0) {
    // Handle initialization error
    return 1;
}
```

Socket Creation

```
SOCKET sock = socket(AF_INET,      // IPv4
                    SOCK_STREAM,   // TCP
                    IPPROTO_TCP);  // Protocol

if (sock == INVALID_SOCKET) {
    WSACleanup();
    return 1;
}
```

Socket Address Structures

```
struct sockaddr_in {
    short            sin_family;   // AF_INET
    u_short         sin_port;     // Port number
    struct in_addr   sin_addr;     // IPv4 address
    char            sin_zero[8];  // Padding
};

// Initialize address structure
struct sockaddr_in addr;
addr.sin_family = AF_INET;
addr.sin_port = htons(8080);          // Port 8080
addr.sin_addr.s_addr = INADDR_ANY;    // Any interface
```

Server-Side Functions

Bind

```
result = bind(sock, (SOCKADDR*)&addr, sizeof(addr));
if (result == SOCKET_ERROR) {
    closesocket(sock);
    WSACleanup();
    return 1;
}
```

Listen

```
result = listen(sock, SOMAXCONN);
if (result == SOCKET_ERROR) {
    closesocket(sock);
    WSACleanup();
    return 1;
}
```

Accept

```
struct sockaddr_in clientAddr;
int clientAddrLen = sizeof(clientAddr);
SOCKET clientSock = accept(sock, (SOCKADDR*)&clientAddr, &clientAddrLen);

if (clientSock == INVALID_SOCKET) {
    closesocket(sock);
    WSACleanup();
    return 1;
}
```

Client-Side Functions

Connect

```
struct sockaddr_in serverAddr;
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(8080);
inet_pton(AF_INET, "127.0.0.1", &serverAddr.sin_addr);

result = connect(sock, (SOCKADDR*)&serverAddr, sizeof(serverAddr));
if (result == SOCKET_ERROR) {
    closesocket(sock);
    WSACleanup();
    return 1;
}
```

Data Transfer Functions

Send/Recv

```
// Send data
const char* sendbuf = "Hello, Server!";
result = send(sock, sendbuf, (int)strlen(sendbuf), 0);
if (result == SOCKET_ERROR) {
    closesocket(sock);
    WSACleanup();
    return 1;
}

// Receive data
char recvbuf[512];
result = recv(sock, recvbuf, 512, 0);
if (result > 0) {
    // Data received, result = number of bytes
} else if (result == 0) {
    // Connection closed
} else {
    // Error occurred
}
```

Non-blocking Sockets

```
// Set non-blocking mode
u_long mode = 1;  // 1 = non-blocking, 0 = blocking
result = ioctlsocket(sock, FIONBIO, &mode);
if (result == SOCKET_ERROR) {
    closesocket(sock);
    WSACleanup();
    return 1;
}
```

Socket Options

Set/Get Socket Options

```
// Enable address reuse
BOOL opt = TRUE;
result = setsockopt(sock, SOL_SOCKET, SO_REUSEADDR,
                    (char*)&opt, sizeof(opt));

// Set timeout
DWORD timeout = 5000;  // 5 seconds
result = setsockopt(sock, SOL_SOCKET, SO_RCVTIMEO,
                    (char*)&timeout, sizeof(timeout));
```

Complete TCP Server Example

```
#include <winsock2.h>
#include <ws2tcpip.h>
#pragma comment(lib, "ws2_32.lib")

int main() {
    WSADATA wsaData;
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {
        return 1;
    }

    SOCKET serverSock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (serverSock == INVALID_SOCKET) {
        WSACleanup();
        return 1;
    }

    struct sockaddr_in serverAddr;
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = INADDR_ANY;
    serverAddr.sin_port = htons(8080);

    if (bind(serverSock, (SOCKADDR*)&serverAddr, sizeof(serverAddr)) == SOCKET_ERROR) {
        closesocket(serverSock);
        WSACleanup();
        return 1;
    }

    if (listen(serverSock, SOMAXCONN) == SOCKET_ERROR) {
        closesocket(serverSock);
        WSACleanup();
        return 1;
    }

    while (true) {
        struct sockaddr_in clientAddr;
        int clientAddrLen = sizeof(clientAddr);
        SOCKET clientSock = accept(serverSock, (SOCKADDR*)&clientAddr, &clientAddrLen);

        if (clientSock != INVALID_SOCKET) {
            char buffer[512];
            int result = recv(clientSock, buffer, 512, 0);
            if (result > 0) {
                send(clientSock, buffer, result, 0);
            }
            closesocket(clientSock);
        }
    }

    closesocket(serverSock);
    WSACleanup();
    return 0;
}
```

Complete TCP Client Example

```
#include <winsock2.h>
#include <ws2tcpip.h>
#pragma comment(lib, "ws2_32.lib")

int main() {
    WSADATA wsaData;
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {
        return 1;
    }

    SOCKET clientSock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (clientSock == INVALID_SOCKET) {
        WSACleanup();
        return 1;
    }

    struct sockaddr_in serverAddr;
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(8080);
    inet_pton(AF_INET, "127.0.0.1", &serverAddr.sin_addr);

    if (connect(clientSock, (SOCKADDR*)&serverAddr, sizeof(serverAddr)) == SOCKET_ERROR) {
        closesocket(clientSock);
        WSACleanup();
        return 1;
    }

    const char* message = "Hello, Server!";
    send(clientSock, message, strlen(message), 0);

    char buffer[512];
    recv(clientSock, buffer, 512, 0);
    closesocket(clientSock);
    WSACleanup();
    return 0;
}
```

Error Handling

```
if (result == SOCKET_ERROR) {
    int error = WSAGetLastError();
    wchar_t* message = NULL;
    FormatMessageW(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
                   NULL, error,
                   MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
                   (LPWSTR)&message, 0, NULL);
    // Use message
    LocalFree(message);
}
```

Best Practices

1. Always initialize WinSock with WSAStartup()
2. Always clean up with WSACleanup() when