**Winsock Exercises 1**

**Exercise 1** (GetHostName)

Task 1: Write a sockets program to get the host name for a given IP address.

Project name: gethostname

Command: gethostname (printing out the host name of your local computer)

Command: gethostname xxx.xxx.xxx.x (printing out the host name of a remote host)

Task 2: Write a sockets program to get the IP address of a given host name.

Project name: gethostaddress

Command: gethostaddress www.sjtu.edu.cn

1. To create a basic Winsock application, create a new empty project with the source file:

#include "pch.h"

#include <winsock2.h>

#include <ws2tcpip.h>

#pragma comment(lib, "Ws2\_32.lib")// Used to link with Ws2\_32.lib

#include <iostream>

using namespace std;

int main(int argc, char \*argv[] ) {

return 0;

}

2. To initialize Winsock and initiate use of WS2\_32.dll:

WORD wVersion = MAKEWORD(2, 2); // Used to request version 2.2 of Windows sockets

WSADATA wsaData; // Data loaded by WSAStartup

int iResult; // Error check if WSAStartup successful

// Initialize Winsock

iResult = WSAStartup(wVersion, &wsaData);

if (iResult != 0) {

cout << "WSAStartup failed: " << iResult << endl;

return 1;

}

3. To close the windows sockets, the following goes at the end of the program:

WSACleanup();

4. To get your local hostname:

char host\_name[128];

gethostname(host\_name, sizeof(host\_name));

cout << host\_name << endl;

5. To retrieve host name for a given IP address passed on the command line:

//To simplify determining buffer requirements for the host and serv parameters, the following values

//for maximum host name length and maximum service name are defined in the Ws2tcpip.h header

//file.

//#define NI\_MAXSERV 32

//#define NI\_MAXHOST 1025

// Validate the parameters

if (argc != 2) {

printf("usage: %s IPv4 address\n", argv[0]);

printf(" to return local hostname\n");

printf(" %s 127.0.0.1\n", argv[0]);

return 1;

}

DWORD dwRetval;

struct sockaddr\_in saGHN;

char hostname[NI\_MAXHOST];

char servInfo[NI\_MAXSERV];

u\_short port = 27015;

//-----------------------------------------

// Set up sockaddr\_in structure which is passed

// to the getnameinfo function

saGHN.sin\_family = AF\_INET;

inet\_pton(AF\_INET, argv[1], &saGHN.sin\_addr.s\_addr);

saGHN.sin\_port = htons(port);

//-----------------------------------------

// Call getnameinfo

dwRetval = getnameinfo((struct sockaddr \*) &saGHN,

sizeof(struct sockaddr),

hostname,

NI\_MAXHOST, servInfo, NI\_MAXSERV, 0);

if (dwRetval != 0) {

printf("getnameinfo failed with error # %ld\n", WSAGetLastError());

return 1;

}

else {

printf("getnameinfo returned hostname = %s\n", hostname);

return 0;

}

6. To retrieve the IP address of a host name passed on the command line:

// Declare and initialize variables

struct addrinfo \*result = NULL;

struct addrinfo \*ptr = NULL;

struct addrinfo hints;

// Validate the parameters

if (argc != 3) {

printf("usage: %s <hostname> <servicename>\n", argv[0]);

printf(" provides protocol-independent translation\n");

printf(" from an ANSI host name to an IP address\n");

printf("%s example usage\n", argv[0]);

printf(" %s www.contoso.com 0\n", argv[0]);

return 1;

}

//--------------------------------

// Setup the hints address info structure

// which is passed to the getaddrinfo() function

ZeroMemory(&hints, sizeof(hints));

hints.ai\_family = AF\_UNSPEC;

hints.ai\_socktype = SOCK\_STREAM;

hints.ai\_protocol = IPPROTO\_TCP;

printf("Calling getaddrinfo with following parameters:\n");

printf("\tnodename = %s\n", argv[1]);

printf("\tservname (or port) = %s\n\n", argv[2]);

//--------------------------------

// Call getaddrinfo(). If the call succeeds,

// the result variable will hold a linked list

// of addrinfo structures containing response

// information

dwRetval = getaddrinfo(argv[1], argv[2], &hints, &result);

if (dwRetval != 0) {

printf("getaddrinfo failed with error: %d\n", dwRetval);

WSACleanup();

return 1;

}

printf("getaddrinfo returned success\n");

// Retrieve each address and print out the hex bytes

struct sockaddr\_in \*sockaddr\_ipv4;

struct sockaddr\_in6 \*sockaddr\_ipv6;

char ipstringbuffer[46];

DWORD ipbufferlength = 46;

int i = 1;

for (ptr = result; ptr != NULL; ptr = ptr->ai\_next) {

printf("getaddrinfo response %d\n", i++);

printf("\tFlags: 0x%x\n", ptr->ai\_flags);

printf("\tFamily: ");

switch (ptr->ai\_family) {

case AF\_UNSPEC:

printf("Unspecified\n");

break;

case AF\_INET:

printf("AF\_INET (IPv4)\n");

sockaddr\_ipv4 = (struct sockaddr\_in \*) ptr->ai\_addr;

inet\_ntop(AF\_INET, &sockaddr\_ipv4->sin\_addr, ipstringbuffer, sizeof(ipstringbuffer));

printf("\tIPv4 address %s\n", ipstringbuffer);

printf("\tport = %d \n ", ntohs(sockaddr\_ipv4->sin\_port));

break;

case AF\_INET6:

printf("AF\_INET6 (IPv6)\n");

// the InetNtop function is available on Windows Vista and later

sockaddr\_ipv6 = (struct sockaddr\_in6 \*) ptr->ai\_addr;

printf("\tIPv6 address %s\n",

inet\_ntop(AF\_INET6, &sockaddr\_ipv6->sin6\_addr, ipstringbuffer, 46) );

printf("\tport = %d \n ", ntohs(sockaddr\_ipv6->sin6\_port));

break;

case AF\_NETBIOS:

printf("AF\_NETBIOS (NetBIOS)\n");

break;

default:

printf("Other %ld\n", ptr->ai\_family);

break;

}

printf("\tSocket type: ");

switch (ptr->ai\_socktype) {

case 0:

printf("Unspecified\n");

break;

case SOCK\_STREAM:

printf("SOCK\_STREAM (stream)\n");

break;

case SOCK\_DGRAM:

printf("SOCK\_DGRAM (datagram) \n");

break;

case SOCK\_RAW:

printf("SOCK\_RAW (raw) \n");

break;

case SOCK\_RDM:

printf("SOCK\_RDM (reliable message datagram)\n");

break;

case SOCK\_SEQPACKET:

printf("SOCK\_SEQPACKET (pseudo-stream packet)\n");

break;

default:

printf("Other %ld\n", ptr->ai\_socktype);

break;

}

printf("\tProtocol: ");

switch (ptr->ai\_protocol) {

case 0:

printf("Unspecified\n");

break;

case IPPROTO\_TCP:

printf("IPPROTO\_TCP (TCP)\n");

break;

case IPPROTO\_UDP:

printf("IPPROTO\_UDP (UDP) \n");

break;

default:

printf("Other %ld\n", ptr->ai\_protocol);

break;

}

printf("\tLength of this sockaddr: %d\n", ptr->ai\_addrlen);

printf("\tCanonical name: %s\n", ptr->ai\_canonname);

}

**Hand in**

1. Your program GetHostName.cpp, GetHostAddress.cpp
2. GetHostInfo.pdf showing (with screen shots)
   1. your local host name and IP address.
   2. the host information for IP addresses xxx.xxx.xxx.xx.
   3. the IP address of given host names (e.g. www.sjtu.edu.cn)
   4. other interesting results you wish to show.