Winsock Exercises 1

GetHostInfo

Contents

- Task 1: gethostname
- Task 2: GetHostAddress

E1: A simple Winsock application

Task1: gethostname

- Write a sockets program to get the host name of a given IP address.
- Project name: gethostname
- Command: gethostname (printing out the host information of your local computer)
- Command : gethostinfo 192.168.100.104 (printing out the host information of a remote host)

```
c:\Winsock\GetHostName\Debug>gethostname
Local Host Name:DELL-JHTang

c:\Winsock\GetHostName\Debug>gethostname 127.0.0.1
Name of remote host = DELL-JHTang
Service of remote host = http

c:\Winsock\GetHostName\Debug>gethostname 192.168.0.154
Name of remote host = X230-TJH.lan
Service of remote host = http

c:\Winsock\GetHostName\Debug>
```

E1 task1 : gethostname

We need to follow these steps

1. Creating a Basic Winsock Application
2. Initializing Winsock
3. Get the host information of a local or remote computer
4. Print the host information
5. Close the Windows sockets

Steps for “Gethostname”

1. Creating a Basic Winsock Application
2. Initializing Winsock
3. Get the host information of a local or remote computer
4. Print the host information
5. Close the Windows sockets

1. Create a basic Winsock Application

- Create a new empty project.
- Add an empty C++ source file to the project.
- Begin programming the Winsock application.

```
#include <winsock2.h>
#include <ws2tcpip.h>
#include <stdio.h>

#pragma comment(lib, "Ws2_32.lib")

int main() {
  return 0;
}
```

The *Winsock2.h* header file contains most of the Winsock functions, structures, and definitions.

Ws2tcpip.h header file contains newer functions and structures used to retrieve IP addresses.

links to the Winsock Library file Ws2_32.lib

Steps for “Gethostname”

1. Creating a Basic Winsock Application
2. Initializing Winsock
3. Get the hostname of a local or remote computer
4. Print the host information
5. Close the Windows sockets

2. Initializing Winsock

– Create a **WSADATA** object called wsaData.

```
WSADATA wsaData;
```

– Call **WSAStartup** and return its value as an integer and check for errors.

```
int iResult;

// Initialize Winsock
iResult = WSAStartup(MAKEWORD(2,2), &wsaData);
if (iResult != 0) {
    printf("WSAStartup failed: %d\n", iResult);
    return 1;
}
```

Steps for "Gethostname"

1. Creating a Basic Winsock Application
2. Initializing Winsock
3. Get the host information of a local or remote computer
4. Print the host information
5. Close the Windows sockets

3. Get the host information – functions and structures

- The **gethostname** function retrieves the standard host name for the local computer.

```
int gethostname(
 _Out_ char *name, //A pointer to a buffer that receives the local host name.
 _In_  int  namelen  //The length, in bytes, of the buffer pointed to by the name parameter.
);
```

If no error occurs, **gethostname** returns zero.

- The **getnameinfo** function provides protocol-independent name resolution from an address to an ANSI host name and from a port number to the ANSI service name..

```
int WSAAPI getnameinfo(
  _In_  const struct sockaddr FAR *sa,      //A pointer to a socket address structure
  _In_  socklen_t                 salen,    //length of sockaddr structure in bytes
  _Out_ char FAR                  *host,    //
  _In_  DWORD                     hostlen,  //
  _Out_ char FAR                  *serv,    //
  _In_  DWORD                     servlen,  //
  _In_  int                       flags     //used to customize processing of the getnameinfo function
);                                          //
```

Steps for “Gethostname”

1. Creating a Basic Winsock Application
2. Initializing Winsock
3. Get the hostname of a local or remote computer
4. Print the host information
5. Close the Windows sockets

4. Print the hostname

```
//----------------------------------------
// Call getnameinfo
          dwRetval = getnameinfo((struct sockaddr *) &saGHN,
                    sizeof(struct sockaddr),
                    hostname,
                    NI_MAXHOST, servInfo, NI_MAXSERV, NI_NUMERICSERV);

          if (dwRetval != 0) {
                    printf("getnameinfo failed with error # %ld\n", WSAGetLastError());
                    return 1;
          }
          else {
                    printf("getnameinfo returned hostname = %s\n", hostname);
                    return 0;
        }
```

5. Close the Windows sockets

```
          WSACleanup();
```

Contents

- Task 1: gethostname
- Task 2: GetHostAddress

E1 task2 GetHostAddress: example output 1

```
c:\Winsock\GetHostAddress\Debug>gethostaddress www.sjtu.edu.cn
usage: gethostaddress <hostname> <servicename>
       provides protocol-independent translation
       from an ANSI host name to an IP address
gethostaddress example usage
   gethostaddress www.contoso.com 0

c:\Winsock\GetHostAddress\Debug>gethostaddress www.sjtu.edu.cn 80
Calling getaddrinfo with following parameters:
        nodename = www.sjtu.edu.cn
        servname (or port) = 80

getaddrinfo returned success
getaddrinfo response 1
        Flags: 0x0
        Family: AF_INET (IPv4)
        IPv4 address 202.120.2.119
        port = 80
        Socket type: SOCK_STREAM (stream)
        Protocol: IPPROTO_TCP (TCP)
        Length of this sockaddr: 16
        Canonical name: (null)
```

E1 task2 GetHostAddress: example output 2

```
c:\Winsock\GetHostAddress\Debug>gethostaddress DELL-JHTang 80
Calling getaddrinfo with following parameters:
        nodename = DELL-JHTang
        servname (or port) = 80

getaddrinfo returned success
getaddrinfo response 1
        Flags: 0x0
        Family: AF_INET6 (IPv6)
        IPv6 address fe80::c10b:5502:31fa:3fb6
        port = 80
        Socket type: SOCK_STREAM (stream)
        Protocol: IPPROTO_TCP (TCP)
        Length of this sockaddr: 28
        Canonical name: (null)
getaddrinfo response 2
        Flags: 0x0
        Family: AF_INET6 (IPv6)
        IPv6 address fe80::a9f7:46da:9992:9dc8
        port = 80
        Socket type: SOCK_STREAM (stream)
        Protocol: IPPROTO_TCP (TCP)
        Length of this sockaddr: 28
        Canonical name: (null)
getaddrinfo response 3
        Flags: 0x0
        Family: AF_INET6 (IPv6)
        IPv6 address fe80::d912:34db:8f45:3034
        port = 80
        Socket type: SOCK_STREAM (stream)
        Protocol: IPPROTO_TCP (TCP)
        Length of this sockaddr: 28
        Canonical name: (null)
getaddrinfo response 4
        Flags: 0x0
        Family: AF_INET (IPv4)
        IPv4 address 192.168.137.1
        port = 80
        Socket type: SOCK_STREAM (stream)
```

E1 task2 : GetHostAddress

We need to follow these steps

1. Creating a Basic Winsock Application
2. Initializing Winsock
3. Get the address information of a local or remote host
4. Print the address information
5. Close the Windows sockets

Steps for E1 task 2

1. Creating a Basic Winsock Application
2. Initializing Winsock
3. Get the address information of a local or remote host
4. Print the address information
5. Close the Windows sockets

3. addrinfo structure and getaddrinfo( ) function

```
typedef struct addrinfo {
  int             ai_flags;      //indicate options used in the getaddrinfo function
  int             ai_family;     //AF_INET or AF_INET6,
  int             ai_socktype;   //SOCK_STREAM，SOCK_DGRAM
  int             ai_protocol;   //IPPROTO_TCP，IPPROTO_UDP
  size_t          ai_addrlen;    //
  char            *ai_canonname; //The canonical name for the host.
  struct sockaddr  *ai_addr;     //A pointer to a sockaddr structure
  struct addrinfo  *ai_next;     //A pointer to the next structure in a linked list
} ADDRINFOA, *PADDRINFOA;
```

```
/* int WSAAPI getaddrinfo (
  _In_opt_       PCSTR      pNodeName,   //host (node) name or a numeric host address string
  _In_opt_       PCSTR      pServiceName, //a service name or port number represented as a string
  _In_opt_ const ADDRINFOA  *pHints, //  provides hints about the type of socket the caller supports
  _Out_          PADDRINFOA *ppResult //  A pointer to a linked list of one or more addrinfo structures
                                      // that  contains response information about the host.
); */
```

`getaddrinfo( ):` Success returns zero. Failure returns a nonzero Windows Sockets error code.

Steps for E1 task 2

1. Creating a Basic Winsock Application
2. Initializing Winsock
3. Get the address information of a local or remote host
4. Print the address information
5. Close the Windows sockets

3. Call **getaddrinfo** function to find the IP address for the **server** name passed on the command line.

```
// 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;

// Call getaddrinfo().
dwRetval = getaddrinfo(argv[1], DEFAULT_PORT, &hints, &result);
    if ( iResult != 0 ) {
        printf("getaddrinfo failed with error: %d\n", dwRetval);
        WSACleanup();
        return 1;
    }
```

Steps for E1 task 2

1. Creating a Basic Winsock Application
2. Initializing Winsock
3. Get the address information of a local or remote host
4. Print the address information
5. Close the Windows sockets

4. Retrieve each address and print out the hex.

```
typedef struct addrinfo {
  int              ai_flags;      //indicate options used in the getaddrinfo function
  int              ai_family;     //AF_INET or AF_INET6,
  int              ai_socktype;   //SOCK_STREAM，SOCK_DGRAM
  int              ai_protocol;   //IPPROTO_TCP，IPPROTO_UDP
  size_t           ai_addrlen;    //
  char             *ai_canonname;//The canonical name for the host.
  struct sockaddr  *ai_addr;      //A pointer to a sockaddr structure
  struct addrinfo  *ai_next;      //A pointer to the next structure in a linked list
} ADDRINFOA, *PADDRINFOA;
```

```
for (ptr = result; ptr != NULL; ptr = ptr->ai_next) {

……

}
```