C++ Easy Socket

Official Documentation

Author(s)

Cryptogenic (Specter)

General Information and Purpose

The “C++ Easy Socket” project was developed to make networking projects easier by hiding all of the messy difficult stuff. It provides a higher-level of abstraction without compromising portability in regards to libraries. It’s currently been built and coded for Linux, however Windows compatibility will come in the near future, as well as support for server-sided functions (eg. binding). If anyone out there wants to help out by implementing either of these, feel free to fork the repo, I may even put it in the official branch!

Motivation

This class was originally implemented for an IRC client I was creating in C++. After seeing all the code and messing around it took to use sockets, I decided to write my own class. After I implemented the program it occurred to me this might be of use to others. It’s by no means perfect, but improvements will be made over time.

Latest Version

The latest version for this project is 1.0. Further implementation such as server support and windows support will be added in the near future.

Requirements

This project does not require any external libraries, all you need is gcc/g++ or any other compiler to compile the source code.

License

C++ Easy Socket is distributed under the Apache License, meaning you are free to modify, distribute, and use for commercial/private use, however the developers of C++ Easy Socket will not be held liable. For more information, please view the Apache v2.0 license before distribution or modification.

Example Usage

To use C++ Easy Socket, it's relatively simple. One just needs to create an instance of the object, initialize it with the init() function, using the host and port to use for parameters. Then the socket can be read or written to via the readSocket() and writeSocket() functions. It is recommended to use checkStatus() to ensure the connection is established before attempting to read or write.

```
int main()
{
  Socket sock;

  std::string host = "127.0.0.1";
  std::string port = "6667";
  sock.init(host, port);

  if(sock.checkStatus())
  {
    std::cout << "The connection was established successfully, terminating...\n";
    return 0;
  }
  else
  {
    std::cout << "The connection could not be established, terminating...\n";
    return -1;
  }
```

Socket Class Outline

Class: Socket

Description: Allows a basic client socket to be opened and contains method for data to be sent and received.

Public Methods:

Socket() – Class constructor, sets status variable to default integer of 1

~Socket() – Class destructor, cleans memory and unsets the socket object

Bool checkStatus() – Returns true if the connection is established

Void init() – Initializes the socket with the given host and port

Void writeSocket – Writes the specified data (argument) to the socket

Void readSocket – Reads and returns data from the socket

Private Methods:

Void throwError() – Prints given error (argument) and sets false status

Void throwWarning() – Prints given warning, but does not set false status

Private Properties:

Int status – Contains the active status code of the socket

Int sockfd – Contains the reference to the socket object as an integer

Int portNumber – Contains the port number for the socket as an integer

Int n – Contains the status code of the socket functions (eg. read, write)

Char buffer – Contains the data for reading

Struct sockadd_in – Contains the family, port, host, zero for socket

Struct serv_addr – Also contains family, port, host.

Struct hostent – Contains host name, aliases, address type, and length.

Additional Information

For additional information on the three private structures, see the TutorialsPoint reference for Unix sockets at http://www.tutorialspoint.com/unix_sockets/socket_structures.htm