Socket

Generado por Doxygen 1.8.4

Jueves, 12 de Diciembre de 2013 17:44:34

Índice general

Capítulo 1

Página principal

La librería Socket es una implementación de comunicación con Sockets TCP/IP en `GNU/Linux`.

El objetivo principal de esta librería es abstraer toda la capa de comunicación al programador, lo único que debe hacer es guardar los datos a enviar en un contenedor STL del tipo `std::string` y enviarlo a una instancia debidamente incializada de la clase Socket. Este se encarga de inicializar los sockets, atender las conexiones o realizar nuevas, etc.

Para más información, consultar el HOWTO

Autor

Imanol Barba Sabariego

Fecha

11/06/2013

Capítulo 2

Client

Ejemplo de aplicación cliente

```
#include "Socket.h"
#include <iostream>
#include "SocketException.h"
#include <sstream>
#include <signal.h>
#include <cstdlib>
#include <sys/time.h>

using namespace std;

void exitClient(int signal)
{
        cout « "Server connection terminated unexpectedly" « endl « "Exiting" « endl;
        exit(-1);
}

int main()
{
        signal(SIGPIPE, exitClient);
        signal(SIGINT, exitClient);
        Socket s;
        string send, recv, host;
        int port;
        s.Create();
        cout « "Created socket" « endl;
        cout « "Hostname: ";
        cin » host;
        cout « "Port: ";
        cin » port;
        cin.ignore();
        s.Connect(host,port);
        cout « "Connected" « endl;
        while(true)
        {
                cout « "> ";
                getline(cin,send);
                s » recv;
                cout « "Received: " « recv « endl;
        }
}
```

Capítulo 3

HOWTO

Descripción de funcionamiento y uso de la librería.

A continuación se desglosan las instrucciones para el desarrollo con esta libería y su posterior uso.

3.1. Descripción del protocolo

En las comunicaciones con Sockets hay un problema, es fácil saber la longitud del mensaje que vas a enviar, pero cuesta saber la longitud del mensaje que deseas recibir, ya que la transmisión se puede detener por llegar al fin del mensaje o bien por que hay problemas en la red.

Los sockets de UNIX nos dan una solución, que es usar llamadas no bloqueantes para recibir el mensaje, es decir: Si leo de un socket, y no he llenado el buffer donde guardo los datos que recibo de allí, la aplicación no se bloquea esperando recibir la suficiente cantidad de datos. A su vez, se usan llamadas a la función `select()` para comprovar si hay datos disponibles a para leer o la conexión está terminada.

Esto requiere uso de la forks e ir haciendo encuesta (polling) al socket, cosa que consume más recursos del sistema.

Debido a la complejidad añadida que supone hacerlo por esta vía, ya que la librería usa threads que son más ligeros para el sistema y la memoria RAM; se ha optado por implementar un sencillísimo protocolo de comunicación que no supone gasto de computación y repercute de forma nímia en el rendimiento de la transferencia.

3.1.1. Modo en texto llano

Este modo transmite los carácteres de texto sin ningún tipo de encriptación, por lo tanto son totalmente visibles para cualquier agente intermedio. Sin embargo, el modo sin encriptación es más rápido y ofrece el doble de velocidad de transferencia aproximadamente.

3.1.2. Envío/Recepción de mensajes

El mensaje se transmite de la siguiente forma:

1. Bob le quiere mandar a Alice un mensaje, primero pone el número de carácteres (bytes) que ocupa el mensaje en forma de string terminado con carácter NULL ('\0').
2. Bob añade el mensaje íntegro y lo manda todo junto.
3. Alice lee uno a uno los carácteres del socket hasta encontrar un carácter NULL ('\0'), entonces lee lo que ha recibido, que es la longitud del mensaje real.
4. Alice pasa a extraer los N bytes que ha leído que recibiría y obtiene el mensaje final.

3.2. Compilación

A continuación se detallan las opciones e instrucciones necesarias para compilar esta librería.

3.2.1. Flags de compilación

Flags de compilación necesarios:

- -lpthread

3.3. Ejemplos

A continuación se detalla el código para programar una aplicación servidor cliente sencilla

3.3.1. Programar un cliente

Véase: Client

Capítulo 4

Índice de clases

4.1. Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

Capítulo 5

Indice de archivos

5.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

Capítulo 6

Documentación de las clases

6.1. Referencia de la Clase Socket

Clase del socket.

`#include <Socket.h>`

Métodos públicos

- Socket ()

  Constructor de la clase Socket.
- void Create ()

  Método para crear sockets.
- void Bind (string address, int port)

  Método para hacer bind a una dirección y puerto.
- void Listen (int backlog)

  Método para escuchar conexiones.
- void Accept (Socket &clientSock)

  Método para aceptar conexiones.
- void Connect (string hostname, int port)

  Método para efectuar conexiones.
- void Close ()

  Método para cerrar sockets.
- int getSock ()

  Getter para el file descriptor del socket.
- const Socket & operator<< (const string &)

  Método para enviar mensajes.
- const Socket & operator>> (string &)

  Método para recibir mensajes.

Métodos privados

- int Receive (char *buff, int length)

  Método para recibir un mensaje de longitud conocida.
- int Send (const char *buff, int length)

  Método para enviar un mensaje de longitud conocida.

Atributos privados

- int sock

  Descriptor del fichero del socket.
- struct sockaddr_in sockAddr

  Estructura de dirección de socket.

6.1.1. Descripción detallada

Clase del socket.

Esta clase define un objeto con los métodos y atributos necesarios para realizar comunicación encriptada o en texto llano a través de un socket TCP/IP en un entorno UNIX abstrayendo la implementación de sockets y de encriptación al programador.

6.1.2. Documentación del constructor y destructor

6.1.2.1. Socket::Socket ( )

Constructor de la clase Socket.

Inicializa el file descriptor del socket y prepara la memoria donde se almacenarán las llaves públicas.

6.1.3. Documentación de las funciones miembro

6.1.3.1. void Socket::Accept (Socket & *clientSock*)

Método para aceptar conexiones.

```
Este método bloquea el thread que lo ejecuta hasta que recibe una conexión entrante, que almacena como instanc
```

Socket en la referencia proporcionada por argumento.

Parámetros

clientSock	Instáncia de Socket que comunica con el cliente entrante

6.1.3.2. void Socket::Bind (string *address*, int *port*)

Método para hacer bind a una dirección y puerto.

```
Este método asigna el socket a una dirección IP (y por tanto, a una interfaz de red en concreto) y un puerto,
```

posteriormente actúe de servidor escuchando conexiones.

Parámetros

address	Dirección IP asignada
port	Puerto asignado

6.1.3.3. void Socket::Close ( )

Método para cerrar sockets.

Este método cierra el socket para que no se pueda escribir ni leer más en él, para liberarlo del kernel y terminar la conexión TCP.

6.1.3.4. void Socket::Connect (string *hostname*, int *port*)

Método para efectuar conexiones.

```
Este método sirve a los sockets que actúan como cliente para poder efectuar conexiones a otro socket que esté
```

dirección y puerto especificados y atienda conexiones.

Parámetros

hostname	Hostname al que conectarse
port	Puerto al que conectarse

6.1.3.5. void Socket::Create ( )

Método para crear sockets.

Crea un file descriptor para un socket que por defecto no está conectado ni asignado a ninguna dirección.

6.1.3.6. int Socket::getSock ( )

Getter para el file descriptor del socket.

Este método devuelve el file descriptor del socket.

6.1.3.7. void Socket::Listen (int *backlog*)

Método para escuchar conexiones.

Este método configura el socket para que se ponga en modo escucha y así pueda atender conexiones entrantes.

Parámetros

backlog	Número máximo de conexiones **en espera**

6.1.3.8. const Socket & Socket::operator<< (const string & *text*)

Método para enviar mensajes.

```
Este método envía el mensaje que se le proporciona a través del Socket con o sin encriptación según
```

las opciones de compilación usando el protocolo implementado.

Parámetros

text	Mensaje a enviar

6.1.3.9. const Socket & Socket::operator>> (std::string & *text*)

Método para recibir mensajes.

```
Este método recibe un mensaje de longitud arbitrária con o sin encriptación según las \ref defines "opciones c
```

el protocolo implementado.

Parámetros

text	Mensaje a recibir

6.1.3.10. int Socket::Receive (char * *buff,* int *length*) [private]

Método para recibir un mensaje de longitud conocida.

Este método se usa para recibir un mensaje de la longitud que se especifica por argumento y almacenarlo en el buffer proporcionado.

Este método garantiza que todo el mensaje se recibirá entero aunque la red no admita una longitud de paquete tan grande.

Parámetros

buff	Buffer donde se almacena el mensaje recibido
length	Longitud del mensaje a recibir

6.1.3.11. int Socket::Send (const char * *buff,* int *length*) [private]

Método para enviar un mensaje de longitud conocida.

Este método se usa para enviar un mensaje de la longitud especifidada en el argumento y devuelve el número de bytes enviados.

Este método garantiza que todo el mensaje se enviará entero aunque la red no admita una longitud de paquete tan grande.

Parámetros

buff	Buffer con el mensaje a enviar
length	Longitud del mensaje

6.1.4. Documentación de los datos miembro

6.1.4.1. int Socket::sock [private]

Descriptor del fichero del socket.

Esta variable contiene el file descriptor del socket abierto por el SO.

6.1.4.2. struct sockaddr_in Socket::sockAddr [private]

Estructura de dirección de socket.

Este struct es usado por el SO para gestionar la dirección del socket abierto.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- /home/imanol/devel/eclipse/PAD/JChatClient/src/include/Socket.h
- /home/imanol/devel/eclipse/PAD/JChatClient/src/Socket.cpp

6.2. Referencia de la Clase SocketException

Clase de Excepción de Sockets.

`#include <SocketException.h>`

Métodos públicos

- SocketException (string m)

  Constructor de la clase.
- ~SocketException ()
- string description ()

  Getter del atributo message.

Atributos privados

- string message

  Mensaje asociado a la excepción producida.

6.2.1. Descripción detallada

Clase de Excepción de Sockets.

Definición de las excepciones lanzadas por la librería de comunicación TCP/IP

6.2.2. Documentación del constructor y destructor

6.2.2.1. SocketException::SocketException (string *m*) [inline]

Constructor de la clase.

Inicializa el mensaje

Parámetros

m	Mensaje inicial

6.2.2.2. SocketException::~SocketException ( ) [inline]

6.2.3. Documentación de las funciones miembro

6.2.3.1. string SocketException::description ( ) [inline]

Getter del atributo message.

Devuelve el mensaje asociado a la excepción producida

6.2.4. Documentación de los datos miembro

6.2.4.1. string SocketException::message [private]

Mensaje asociado a la excepción producida.

La documentación para esta clase fue generada a partir del siguiente fichero:

- /home/imanol/devel/eclipse/PAD/JChatClient/src/include/SocketException.h

6.3. Referencia de la Estructura thread_args

Argumentos de los threads.

`#include <client.h>`

Atributos públicos

- pthread_mutex_t * mutex

  Variable de control de la exclusión mútua entre threads.
- pthread_cond_t * condition

  Variable de notificación a otros threads.
- Socket * s

  Puntero al socket.

6.3.1. Descripción detallada

Argumentos de los threads.

Este struct define los argumentos que recibe un thread abierto por la aplicación servidor al recibir una conexión entrante

Autor

Imanol Barba Sabariego

Fecha

11/06/2013

6.3.2. Documentación de los datos miembro

6.3.2.1. pthread_cond_t* thread_args::condition

Variable de notificación a otros threads.

Esta variable se usa para notificar a otros threads cuando deben realizar otras acciones.

Actualmente se usa para notificar al thread principal cuando el thread que lo invoca ha terminado, en caso de que el principal haya quedado bloqueado y no admita más conexiones.*

6.3.2.2. pthread_mutex_t* thread_args::mutex

Variable de control de la exclusión mútua entre threads.

Esta variable se usa para bloquear otros threads en operaciones de exclusion mútua donde se modifican variables compartidas

6.3.2.3. Socket* thread_args::s

Puntero al socket.

Esta variable representa el puntero al socket que proviene de la conexión entrante recibida por el servidor. Con este, el thread puede recibir y enviar los datos.

La documentación para esta estructura fue generada a partir del siguiente fichero:

- /home/imanol/devel/eclipse/PAD/JChatClient/src/include/client.h

Capítulo 7

Documentación de archivos

7.1. Referencia del Archivo /home/imanol/devel/eclipse/PAD/JChatClient/src/client.cpp

Fichero de implementación de un cliente.

```
#include "client.h"
```

Funciones

- void killThread (thread_args *t_arg)

  Método de liberación de los argumentos.
- bool connect (Socket &s)

  Método de conexión.
- void * sendThread (void *args)

  Thread de envío de mensajes.
- void * recvThread (void *args)

  Thread de recepción de mensajes.
- int main ()

  Método principal del cliente.

Variables

- bool connected

  Variable de estado del programa.
- bool finished

  Variable de estado del programa.

7.1.1. Descripción detallada

Fichero de implementación de un cliente.

Autor

Imanol Barba Sabariego

Fecha

13/06/2013

En este fichero se implementa un cliente para poder usar con el servidor creado, usando la clase Socket. REVISADO EL 10/12/2013 PARA LA APLIACIÓN JCHAT

7.1.2. Documentación de las funciones

7.1.2.1. bool connect (Socket & *s*)

Método de conexión.

Este método conecta el cliente a una sala de chat.

7.1.2.2. void killThread (thread_args * *t_arg*)

Método de liberación de los argumentos.

Este método se encarga de liberar la memoria asignada a los argumentos pasados a los threads que ahora han terminado.

7.1.2.3. int main ( )

Método principal del cliente.

Este método inicializa el Socket, establece la conexión y realiza las acciones que se le hayan programado para comunicarse con el servidor.

7.1.2.4. void* recvThread (void * *args*)

Thread de recepción de mensajes.

Este método es ejecutado por un thread con el objetivo de recibir los mensajes por el socket, gestionarlos e imprimirlos por pantalla o tomar las medidas necesarias .

7.1.2.5. void* sendThread (void * *args*)

Thread de envío de mensajes.

Este método es ejecutado por un thread con el objetivo de enviar los mensajes y comandos que el usuario introduce por teclado.

7.1.3. Documentación de las variables

7.1.3.1. bool connected

Variable de estado del programa.

7.1.3.2. bool finished

Variable de estado del programa.

7.2. Referencia del Archivo /home/imanol/devel/eclipse/PAD/JChatClient/src/client.dox

7.2.1. Descripción detallada

Autor

Imanol Barba Sabariego

Fecha

13/06/2013

7.3. Referencia del Archivo /home/imanol/devel/eclipse/PAD/JChatClient/src/howto.dox

7.3.1. Descripción detallada

Autor

Imanol Barba Sabariego

Fecha

13/06/2013

7.4. Referencia del Archivo /home/imanol/devel/eclipse/PAD/JChatClient/src/include/client.h

```
#include "Socket.h"
#include <iostream>
#include "SocketException.h"
#include <sstream>
#include <signal.h>
#include <cstdlib>
#include <sys/time.h>
```

Clases

- struct thread_args

  Argumentos de los threads.

7.5. Referencia del Archivo /home/imanol/devel/eclipse/PAD/JChatClient/src/include/Socket.h

Header de la clase Socket.

```
#include <iostream>
#include <sstream>
#include "SocketException.h"
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netinet/tcp.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <cstdlib>
#include <cstring>
```

Clases

- class Socket

  Clase del socket.

7.5.1. Descripción detallada

Header de la clase Socket.

Autor

Imanol Barba Sabariego

Fecha

12/06/2013

En este fichero se define la clase Socket, que es la clase que se abstraer toda la comunicación con sockets al programador

7.6. Referencia del Archivo /home/imanol/devel/eclipse/PAD/JChatClient/src/include/-SocketException.h

Header de excepción de Sockets.

```
#include <string>
```

Clases

- class SocketException

  Clase de Excepción de Sockets.

7.6.1. Descripción detallada

Header de excepción de Sockets.

Autor

Imanol Barba Sabariego

Fecha

10/06/2013

En este fichero se define la clase SocketException para el control de excepciones producidas por la librería.

7.7. Referencia del Archivo /home/imanol/devel/eclipse/PAD/JChatClient/src/mainpage.dox

7.8. Referencia del Archivo /home/imanol/devel/eclipse/PAD/JChatClient/src/makefile.dox

7.9. Referencia del Archivo /home/imanol/devel/eclipse/PAD/JChatClient/src/Socket.cpp

Fichero de implementación de la clase Socket.

```
#include "Socket.h"
```

7.9.1. Descripción detallada

Fichero de implementación de la clase Socket.

Autor

Imanol Barba Sabariego

Fecha

13/06/2013

En este fichero se implementan los métodos de la clase Socket definidos en Socket.h

Índice alfabético